CRTypist: Simulating Touchscreen Typing Behavior via Computational Rationality

Danqing Shi
Aalto University
Finland

Yujun Zhu
Aalto University
Finland

Jussi P.P. Jokinen
University of Jyväskylä
Finland

Aditya Acharya
University of Birmingham
United Kingdom

Aini Putkonen
Aalto University
Finland

Shumin Zhai
Google
United States

Antti Oulasvirta
Aalto University
Finland

ABSTRACT

Touchscreen typing requires coordinating the fingers and visual attention for button-pressing, proofreading, and error correction. Computational models need to account for the associated fast pace, coordination issues, and closed-loop nature of this control problem, which is further complicated by the immense variety of keyboards and users. The paper introduces CRTypist, which generates human-like typing behavior. Its key feature is a reformulation of the supervisory control problem, with the visual attention and motor system being controlled with reference to a working memory representation tracking the text typed thus far. Movement policy is assumed to asymptotically approach optimal performance in line with cognitive and design-related bounds. This flexible model works directly from pixels, without requiring hand-crafted feature engineering for keyboards. It aligns with human data in terms of movements and performance, covers individual differences, and can generalize to diverse keyboard designs. Though limited to skilled typists, the model generates useful estimates of the typing performance achievable under various conditions.

CCS CONCEPTS

• **Human-centered computing → User models**.

KEYWORDS

Simulation models; Reinforcement learning; Touchscreen typing; Computational modeling

ACM Reference Format:
Danqing Shi, Yujun Zhu, Jussi P.P. Jokinen, Aditya Acharya, Aini Putkonen, Shumin Zhai, and Antti Oulasvirta. 2024. CRTypist: Simulating Touchscreen Typing Behavior via Computational Rationality. In *Proceedings of the CHI Conference on Human Factors in Computing Systems (CHI '24), May 11–16, 2024, Honolulu, HI, USA.* ACM, New York, NY, USA, 17 pages. https://doi.org/10.1145/3613904.3642918

CHI '24, May 11–16, 2024, Honolulu, HI, USA

ACM ISBN 979-8-4007-0330-0/24/05
https://doi.org/10.1145/3613904.3642918

1 INTRODUCTION

How might computational models help us improve text entry methods for human use? Since the field's inception in the 1980s, Human-Computer Interaction (HCI) has been in pursuit of models that could illuminate and predict outcomes in typing [15]. Some models predict performance in elementary subtasks, such as selecting keys [12], or visually searching them [44], and choice [53], while some cover compound tasks, such as transcription typing [14, 42, 43, 77]. Successful models would have multiple applications. They could advance the formation of theories of typing and provide insights into usability of prototypes before user testing, and thereby lower the costs of human-centric engineering. They could help improve the accessibility of interfaces by simulating users with different capabilities, which might be hard or impossible to recruit (e.g., [80]). They could be applied to algorithmically design better layouts and drive decoding algorithms that better adapt to users' movement strategies [29, 92]. Finally, models could serve to support machine learning -based methods by generating datasets using simulations of human behavior [56, 57].

However, one critical challenge that stands unsolved is predicting the effects of changing conditions. To illustrate the importance of this point, imagine Alex, a middle-aged user who types comfortably at around 28 words per minute using one finger on a smartphone:

- Change in performance objectives: How would performance and strategies change if Alex aimed to be more careful and leave fewer errors?
- Change in typing style: What if Alex wants to type faster with two thumbs?
- Change in design: Would switching from QWERTY to an optimized keyboard layout improve Alex's performance?
- Change in assistive features: If auto-correction was turned on, would Alex's performance improve or worsen? Would the effectiveness of this feature matter much?
- Change in capabilities: How might deterioration in Alex's memory abilities affect typing?

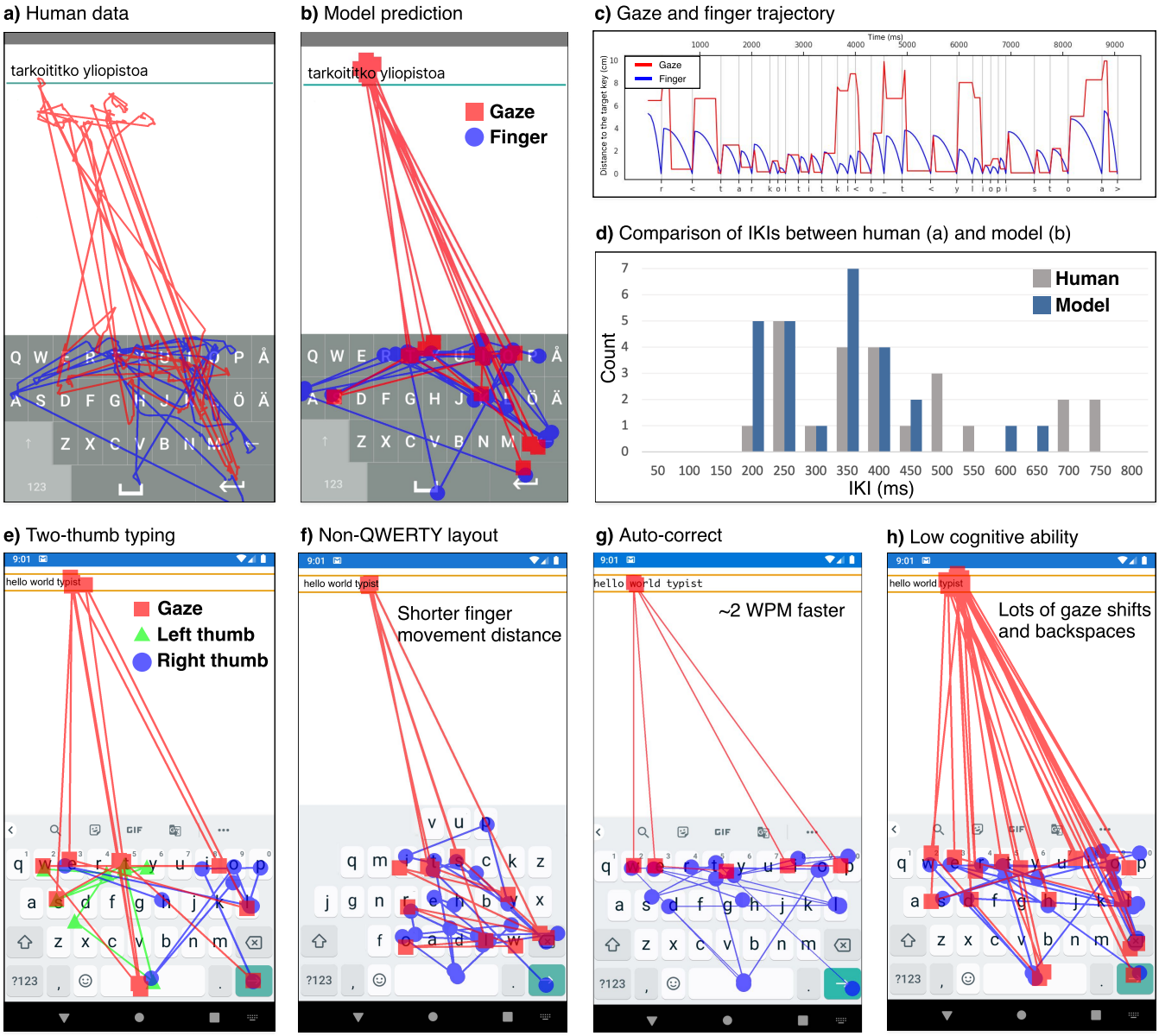

Figure 1: CRTypist predicts typing behavior with touchscreen keyboards, running directly from screenshots or software emulators. The simulated finger- and eye-movement trajectories (a) match humans' (b), including how people shift their attention and correct mistakes (c). Also, the distribution of internal-key intervals (IKIs) produced via CRTypist is similar to the human pattern (d). The model adapts its typing behavior to new conditions by reoptimizing its policy; the examples here reflect two-thumb typing (e), typing with a non-QWERTY layout (f), use of auto-correction (g), and effects of low cognitive ability (h).

Such *counterfactual questions* are claimed to be at the heart of design [62]. However, available models fare poorly in answering them. They do not include mechanisms that enable them to leap from the set of observations they have been trained on or fitted to. For example, models based on Fitts' law require reparameterizing regression coefficients (a and b), which rely on experimental data [12]. Likewise, cognitive-architecture models such as GOMS and EPIC, and their simplified keystroke-level versions, require handcrafting a new production system, which demands deep empirical insight [36, 42]. Could we machine-learn our way out this? No, not without unprecedented data collection effort. Training datasets typically offer a "snapshot" of user behavior particular to a given device and skill level, yet supervised-learning models struggle with out-of-distribution samples. Although current reinforcement learning -based models can learn policies without human data, existing

typing models [43] have limitations when testing their ability to generalize to out-of-distribution samples. This is because they rely on manually crafted representations of the state and action spaces for any new environment, which we aim to address.

In this paper, we introduce CRTypist (Computationally Rational Typist), a computational model that simulates typing behaviors on touchscreens under various conditions (Figure 1). Given a text phrase (*target*) and a virtual keyboard (*design*), CRTypist moves the gaze and taps on a touchscreen. It does so in a human-like manner: it makes mistakes and looks up and down the display to detect and correct them. Because it can "see" and "touch" a screen, the researcher does not need to represent the world for it.

To enable CRTypist, we describe a modular and hierarchical model architecture (Figure 2) that supports simulating touchscreen typing and runs directly on screenshots or emulators. The model builds on the theory of computational rationality [63], and extends it as a pixel-based agent. Computational rationality as a theory of interaction predicts that typing behavior can be predicted as optimization within relevant bounds, in particular the design, the user's capabilities (motor, perceptual, and cognitive), and goals (speed vs. accuracy). The key enabler of the architecture is the supervisory control that a supervisor interacting with an internal environment (cognition), bridging the controller and the touchscreen via a vision module and a finger module. Moreover, it maintains a working memory representation of what it has typed so far. These modules can be trained from pixels to perform pixel-level observation and interaction, which makes the typing simulation more realistic and allows the model to transfer its typing ability to various keyboards regardless of design. The approach goes beyond the previous model [43] in terms of generalizability by incorporating modules trained to work directly from pixels, instead of task-oriented modules like pointing or proofreading.

We report results from a series of experiments. Using comparison with the baseline approach [43] and empirical data [41], we assessed CRTypist's ability to reproduce human-like one-finger and two-thumb typing behavior. The results indicate that CRTypist performs comparably to, or even more accurately than, the baseline technique regarding typing speed, error correction, and proofreading strategy. Next, exploring the range of typing behavior that our model can address, the model successfully predicted performance across a spectrum of human capabilities. Last, we tested the generalizability of the agent by evaluating CRTypist for real-world keyboard screenshots, and examining how the model captures typing behaviors arising from novel keyboard layouts and an auto-correct feature. The experiments' results highlight the potential that CRTypist offers for real-world environments and showcases its applicability for predicting how human cognition adapts to keyboards with different features. The model is limited to the prediction of performance after practice; it does not yet model skill acquisition.

Our main contribution is a computational model of typing that, for the first time, enables accurate simulation of typing in terms of typing speed and strategies for error correction and proofreading directly from pixels without hand-crafted state representations. The model reproduces a broad range of empirical phenomena in touchscreen typing, including one- vs. two-thumb typing, individual differences, the effects of keyboard design, as well as that of the auto-correction feature. The core technical innovation is a reformulation of the supervisory control problem: here, visual attention and fingers are controlled based on working memory, which continuously updates a time-decaying belief about what has been typed so far. Our architecture is modular and hierarchical: the vision, finger, and working memory modules are trained to a sufficient level of competence, after which they can be utilized with the supervisor. This approach allows flexibility critical for practitioners: The pre-trained model can be run in conditions and on keyboards not contained in the dataset. To facilitate the training and evaluation of this new breed of models, we release a benchmark for touchscreen typing. We also opensource the model that can be downloaded by others and used for evaluation, design, and engineering [1].

2 BACKGROUND

This section reviews background knowledge about human touchscreen typing behavior and points out the research gap in the current modeling approaches.

2.1 Human Typing on Mobile Touchscreens

How we type. Typing is a complex process involving numerous cognitive, perceptual, and motor abilities [78]. Multiple physical and cognitive constraints of human typists come into play: 1) The finger-movement inaccuracies and ambiguities caused by the inherently noisy motor actions [64] necessitate a tradeoff between achieving quick input with greater potential for error and achieving accuracy at the expense of speed [30]. 2) The human visual system has limited information-processing capacity, and only a tiny foveated area of the visual field can be in sharp focus at any given time [17]. 3) Information held in working memory will likely decay over time [72], requiring one to look at the text display to reduce uncertainty about what was typed. 4) Virtual keyboards on mobile touchscreens lack the tactile feedback of physical keyboards [35]. 5) Most modern virtual keyboards, such as the iOS keyboard [84] and Google's Gboard [55], offer auto-correction, completion, and prediction functions that may demand additional attention and selection actions from users.

The combination of these phenomena makes the typing behavior complicated. In touchscreen typing, utilizing the limited resources proves especially challenging because of a conflict rooted in the inherent constraints of human vision [79]: on one hand, the variability in finger movements necessitates continuous visual guidance due to the unpredictability of motor responses [41], yet the same visual attention must be allocated to proofreading the typed text [74]. Users have to divide their visual attention between guiding the motions on the keyboard and proofreading the text entered. Hence, text entry is an optimization challenge for the typist: what is the correct ratio for gazing at the text vs. the keyboard, and how frequently should the gaze shift? A recent study [41] reports that the typists kept their gaze on the keyboard around 70% of the time when entering text with one finger and 60% when using their thumbs. Glances at the text, for proofreading, were interleaved with typing, with roughly four such glances per sentence with a mean length of 20 characters. While this insight is valuable for understanding human hand-eye coordination strategies, it is only a start.

[1] https://crtypist.github.io

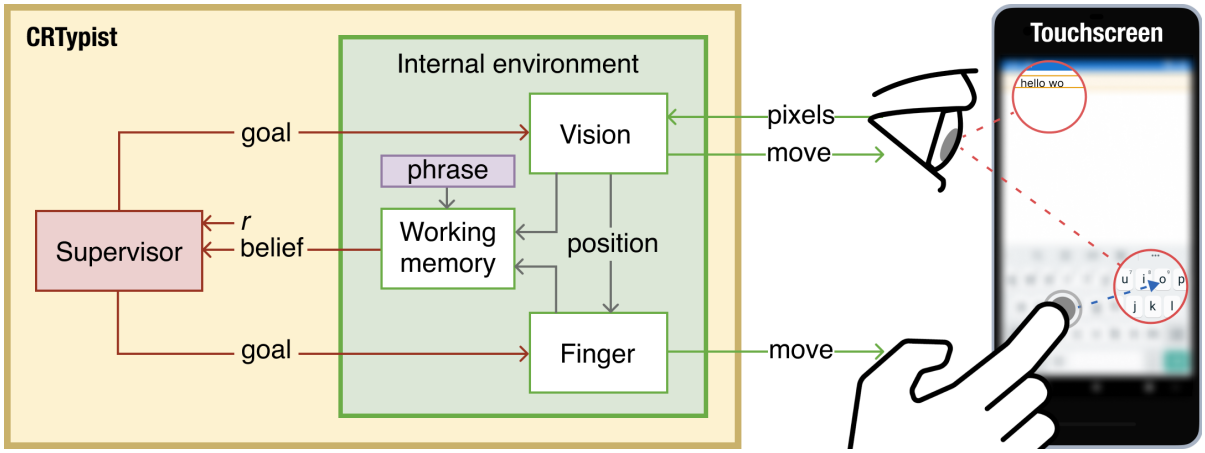

Figure 2: An overview of the modeling approach. The supervisory control problem is modeled as deciding where to look next and where to move the finger next. The supervisor does not have direct access to the state of the touchscreen – it must rely on an internal environment, which bridges the supervisory controller and touchscreen in conditions of cognitive and physical limitations. Within the internal environment, there are three internal modules: vision moves the gaze and observes the screen from pixels through foveated and peripheral vision; finger obtains a position accordingly and taps on the virtual keyboard; and working memory infers what has been typed thus far, updating the related beliefs, by means of the target text phrase and information from vision and finger. At high level, the supervisor reads working memory and sets goals for vision and finger. The reward r is defined with a speed–accuracy tradeoff: typing correct target phrases as quickly as possible.

How individual differences affect typing. Beyond population-level patterns, typing performance and strategies are influenced by large individual-level differences. The range of typing speeds typically reported, 25–40 words per minute (WPM), points to a wide range of performance capacities among touchscreen typists [66]. Interestingly, while two-thumb typing has been associated with higher speeds, it also results in a higher error rate [5, 60]. Despite this drawback and the increased muscle activity entailed, most individuals generally favor two-thumb typing unless the context or device creates constraints that limit its application [90]. Another pivotal determinant of typing speed is the user's familiarity with the layout; novices, lacking layout knowledge, often underperform, with speeds as low as 7 WPM, while experts can reach speeds upwards of 29 WPM in controlled settings [40, 44], and as high as 80 WPM in naturalistic ones [66]. Typists also adjust their typing preferences to the situation. When they are free to leave errors in mobile emails [85], they backspace approximately twice per sentence [66]. In contrast, when there is greater emphasis on error avoidance, the frequency of using the Backspace key inevitably rises [41].

How the keyboard design influences typing. The design of touchscreen keyboards has a significant impact on typing behavior. Key size plays a crucial role in improving typing speed and reducing the error rate in touchscreen typing, as Parhi et al. have shown [67]. Layout too is vital, as studies of optimal stylus-keyboard layouts shows [92]: across QWERTY, CHUBON, FITALY, and OPTI keyboards, speeds ranged from 30 to 38 words per minute. In one optimization effort, Oulasvirta et al. [65] designed a new split keyboard layout for two-thumb fast text entry, which minimized thumb-travel distance and maximized alternation between thumbs. In addition to layout, designers frequently employ intelligent text-entry features to enhance efficiency. For instance, an effective auto-correct feature can reduce the time required to manually correct typing errors and raise typing speeds [9]. Typists speed up and move more quickly between keys when less concerned about errors [7, 8]. Hence, accurate word suggestions have been shown to improve both user satisfaction and typing speed [76].

2.2 The Research Gap

Typing was long modeled purely as motor performance [77]. The impact of keys' size and relative positioning can be effectively predicted via classic approaches such as Fitts' law [12]. However, these approaches disregard the interaction of human vision and motor system. Although rule-based models such as ACT-R [1] and EPIC [48] can use step-by-step simulation for both eye and finger movements, there are no comprehensive text entry models built on ACT-R or EPIC that consider proofreading and errors. Until recently, the latest study [43] has yielded a computational model for predicting both eye and finger movement via supervisory optimal control, which helps shed light on error correction and proofreading. Though valuable, this state-of-the-art approach still has two major limitations. Firstly, its fixed parameters and task-specific design (for pointing and proofreading) render it unable to simulate diverse behaviors and thus reflect a wide range of individual users. While an ability-based optimization approach [81] might assist in this regard, that still cannot provide a training workflow to integrate with the optimization pipeline. Secondly, generalizability to typing behaviors with various real-world keyboards is lacking, because the existing model uses hand-crafted state–action spaces representing the environment. This limits its real-world usability.

To summarize, existing approaches to modeling touchscreen typing do not support eye–hand movements' simulation, handling

of individual-level factors, and generalizability at the same time. Generalizability related to keyboard designs remains challenging.

3 CRTYPIST: MODELING TOUCHSCREEN TYPING

We adopted three strategies to design the computational model for generating typing behavior from pixels. These strategies embrace the foundational theory of *computational rationality* and two overarching design considerations – namely, *hierarchical supervisory control* and *modular architecture*.

- **Computational rationality**. The premise of computational rationality posits that humans adopt behaviors that maximize expected utility within given bounds [63]. In our design, these constraints, or bounds, are categorized as external (pertaining to the operation environment) and internal (related to cognitive factors such as visual perception, motor control, and working memory) environment. The supervisory controller does not interact with the screen pixels directly; instead, it focuses on the internal environment representing the intricate interplay of visual and motor coordination during typing.
- **Hierarchical supervisory control**. Hierarchical supervisory control denotes a tiered control system where superior modules set goals for their subordinates. Each sequence of actions from subordinates is integrated into an overall pattern for the higher-level control [71]. Research suggests that humans utilize such hierarchical frameworks to manage the multifaceted challenges encountered in real-world settings, aiding both learning and decision-making [13, 27]. For instance, the role of vision in the hierarchical organization is to help its supervisor observe pixels and guide attention in searching for the text display and keys [89]. Hierarchical architectures are pivotal in machine learning, especially for breaking intricate tasks down into simpler subtasks [10].
- **Modular architecture**. Modularization divides a system into distinct components that function autonomously but can collaborate to reach broader objectives. This principle fosters adaptability and architectural clarity while speeding up the development process [68]. There is evidence that the human cognitive system is modular in nature, with separate modules handling specific cognitive tasks [1, 25]. By adopting a modular architecture, we can adjust modules to imitate certain human abilities. Such modularization has been adopted in machine learning to enhance the efficiency and performance of deep-learning models [3].

Two fundamental principles lie behind these strategies: One is the principle of parsimony ("*Entities are not to be multiplied without necessity*", William of Occam), which suggests choosing the least complex model that describes the data well [58]. Given the problem space of modeling, we employed only a small number of modules and kept each one as simple as possible. The other principle is glass-box modeling, a fundamental principle behind explainable artificial intelligence [32]. Model transparency helps researchers to understand and access the decision-making processes of the model. Such a level of transparency requires that the model structures' and modules' design be consistent with cognitive-science research.

3.1 Problem Formulation

Generating typing behavior on touchscreens is a sequential decision-making problem to control gaze and finger over time. To represent this user's decision problem accurately, we fomulate it as a bounded optimality problem in a partially observable Markov decision process [83]. The typist observes the state $s \in \mathcal{S}$ through $o \in \mathcal{O}$ and performs an action $a \in \mathcal{A}$ to type target phrases. Given the action $a \in \mathcal{A}$ and current state $s \in \mathcal{S}$, the environment (touchscreen device) gets transitioned to a new state $s' \in \mathcal{S}$. The typist receives a reward r at the end of a typing episode. The following description details the individual terms for capturing the typing task:

- $\mathcal{S}$ is the state space, in which a state s_t is the pixel representation of the touchscreen display at timestep t, including both keyboard and text area.
- $\mathcal{O}$ is the observation space, in which an observation o_t is the information that can be observed by the typist within the cognitive process.
- $\mathcal{A}$ is the action space, in which an action a is behavior the typist can execute, ranging from gaze movements to screen taps.
- r is the reward, which provides feedback from the environment. The reward for typing is defined with a speed–accuracy tradeoff: the goal is to type correct target phrases as quickly as possible.

3.2 Model Design

Drawing from our guiding strategies and the defined problem space, we conceptualize a model premised on human bounds (Fig. 2). Instead of directly interacting with the external environment (touchscreen), the typist's central controller (supervisor) communicates with an intermediary internal environment molded by human cognitive processes. This internal environment, bridging the typist and touchscreen, generates typing behaviors congruent with human cognitive and physical capibilities and limitations. Within this internal environment, we incorporate three pivotal modules: vision, finger, working memory. Following Occam's razor, we aim to make each component model as simple as possible. Each module's design and functionality are elaborated upon in subsequent subsections.

3.2.1 Supervisor. Functioning as the system's central controller, the supervisor manages the internal environment. It takes in the belief from working memory as well as the θ parameters of the internal environment to decide the next action. We consider three parameters in the internal environment: E_K indicates the encoding time of vision, F_K represents the accuracy of finger, and λ determines the capability of working memory. They will be introduced in corresponding component parts.

In each typing episode, the policy $\pi(\theta)$ of the supervisor decides where to look next and where to move the finger next based on the information retrieved from working memory. Specifically, it decides when to deploy the vision for proofreading and when the gaze should be on the keyboard to guide finger movements. It also commands the finger to tap a letter key to complete a phrase or tap backspace for error correction, while also selecting the speed for the finger movement. The gaze and finger movement occur in parallel as the supervisor instruct vision and finger concurrently

in every timestep. Although the supervisor sets goals for both vision and finger movements concurrently in each timestep, it is important to note that finger and eye movements may not occur simultaneously. If finger is already performing an action in the current timestep, vision can still start moving, and the command to finger is ignored. To ensure a high resolution of simulation, we have set the timestep to 50ms, which is comparable to the time it takes for the eye movement and shorter than finger movement. When the "Enter" key is pressed, the typing session ends and the model receives a reward based on accuracy and time efficiency:

$$r = w_1 \cdot (1 - cer^{\alpha}) - w_2 \cdot t \tag{1}$$

The first term is a positive reward for accuracy, where *cer* is the character error rate [52], indicating the percentage of incorrectly typed characters. It has an exponential power constant α to encourage correctness. The second term is a constant penalty over time, where t is the time duration normalized by the length of sentence text. w_1 and w_2 are weights to ensure the speed–accuracy tradeoff.

3.2.2 Vision. The vision component interfaces with the external world and is critical for interactive typing. It is responsible for guiding fingers on the keyboard (since the touchscreen keyboard provides no physical feedback) and reading of the text display for proofreading. Because of limits inherent to human vision, such as the high acuity foveal vision being size-limited, the typist needs to move the gaze to gather accurate visual information about the environment's state. It includes two pixel-based modules to "see" the world: (*foveal* and *peripheral* vision, which align with the bounds imposed on the human visual system [22]) and one policy module to control the gaze position.

- Foveal vision module involves a small high-resolution area of the eye's retina, which can capture information with high visual acuity. When the gaze is on the keyboard, foveal vision can help one see which key it has rested upon; when the gaze is on the text display, the user can read what has been typed there. In the implementation, we first use deep learning-based OCR [54] to recognize text from the pixels of the original screenshot. Then, it crops an image patch (64×64) as the foveal area from the screenshot (256×455) to identify the text and keys present in the patch. The size of the foveal area is approximated when the visual angle is 2 degrees and the distance from the touchscreen is about 16 inches.
- Peripheral vision module, in turn, refers to what is outside the center of foveal vision. While visual acuity and detail perception are reduced in peripheral vision relative to foveal vision, it still provides valuable spatial information about the general layout and arrangement of objects in the environment. In the implementation, the module takes a blurred image of the entire screen as input and uses a CNN-based autoencoder [34] to encode it into a dense vector. This vector represents the overall visual information that can reconstruct the original pixels. Since we do not consider switching the keyboard while typing, the peripheral information remains constant and does not change over time.
- Policy module is the controller for the gaze movement. It takes in the command of a goal (e.g., look at key "a") as a one-hot vector and outputs the next coordinate position of gaze. It takes the peripheral vision into account as well for making gaze movement decision, and uses foveal vision to ensure it arrives at the correct place. The policy is modeled by a neural network and trained with reinforcement learning. The reward function for vision is $r = 1-(\frac{d}{d_{max}})^{0.4}$, where d is the distance from the current position to the center of target position, while d_{max} represents the maximum distance based on the size of the keyboard.

We compute the vision's fixation time via the EMMA model [79]. A visual encoding time T_{enc} is included, for the duration of encoding a target, such as a key or a typed word.

$$T_{enc} = E_K \cdot [-\log f_i] \cdot e^{k\varepsilon_i} \tag{2}$$

where f_i represents the frequency of the target being encoded and ε_i denotes the target's eccentricity. The constants E_K and k influence the scaling of the encoding time and the exponent. We use E_K as a free parameter that can express the vision's capability level. A larger E_K imposes a need for more time for visual encoding. Besides the fixation time, we set the time to prepare and execute a saccade for gaze movement to approximately 200 ms [79].

3.2.3 Finger. The finger module emulates the motor control of finger movement, simulating physical touchscreen interactions. Similar to the policy module of vision, the controller of finger is modeled as a neural network that takes in the goal (e.g., tap key "a") as a one-hot vector and outputs the coordinate position of the fingertip to tap the touchscreen. In addition to its goal, the finger also utilizes visual information from the peripheral vision to determine its movement based on the keyboard's visual features. The reward function used in finger is identical to the vision policy, with the highest reward being given when finger taps the center of the target.

Movement accuracy, which varies with speed and distance [24], is simulated in the model via the parameterized Weighted Homographic model [31]:

$$(y - y_0)^{1-k_\alpha}(x - x_0)^{k_\alpha} = F_K \tag{3}$$

where x is the movement time of finger. y is the standard deviation for the finger's endpoint spread. F_K is a free parameter that can be adjusted to address finger capability (a smaller F_K represents more accurate finger movement). This model encourages slow movement for an accurate result. Furthermore, the error gets worse if the finger lacks guidance from vision. Our method adds increasing Gaussian-distributed noise to the finger's position over the duration without visual guidance.

3.2.4 Working Memory. Finally, working memory serves as a temporary mental storage system that holds information briefly [26, 49, 50]. It is fundamental to the task of typing, not only storing information but also processing it. Specifically, during the process of deciding what to type next, the model preserves details about the target phrase, the text already typed, and its correctness. On the other hand, the purpose of modeling working memory is not only remembering all that has been typed, but also simulating human's limitation in recall. Specifically, working memory encapsulates the uncertainty about what has been typed [37]. During typing, the correctness and certainty of the text stored in the working memory changes over time, requiring proofreading to ensure accuracy and

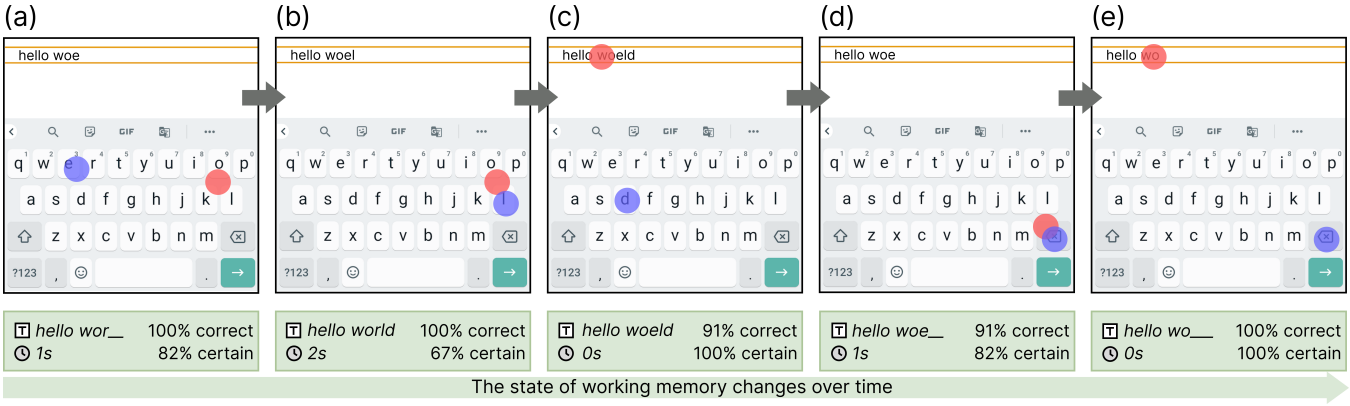

Figure 3: An example of how the model's internal working memory module helps the agent track what has been typed as the observations and agent actions progress. Here, the module (WM, $\lambda = 0.2$) updates itself as the model types "hello world" on a touchscreen. The green box indicates the information held by WM: the typed text stored 🅃, time since last proofreading 🕓, and corresponding correctness and certainty. The panes illustrate WM missing an error (tapping e instead of r) as the vision fixates on neither the finger's position nor the text display (a); the phrase in WM being 100% correct but certainty decreasing rapidly if one does not proofread for a long time (67%) (b); low certainty of correctness from WM steering the supervisor toward proofreading, which helps identify the error (c); upon identification, vision guiding the finger to make multiple Backspace presses (d); and the model proofreading the text again and performing one more backspace to correct the remaining error (e).

efficiency. We designed working memory to follow three stages accordingly [11, 69, 70]:

- *Encoding and Storage.* The module encodes information received from the vision and finger movements into the typed text and stores it in memory. When vision is proofreading, it uses the text from the display as the input text. When vision is on finger's position, a neural network, which predicts the typed text from vision's feature and finger's position, is trained through supervised learning for encoding movement details into text. When vision is on keyboard but not on finger's position, it directly uses the goal of finger as the input character.
- *Maintenance.* The module maintains the correctness and certainty of the typed text. It measures the correctness of the typed text $P_{correctness} = 1 - cer$ in memory based on the character error rate cer [52]. As time progresses, the certainty of information stored in memory decreases. We apply a decay model $P_{certainty} = e^{-\lambda\tau}$ to simulate this process, where τ is the time elapsed since the last proofreading, and λ is the parameter for tuning the capability. A lower λ indicates a stronger level of certainty, and $\lambda = 0$ denotes perfect confidence in all time.
- *Retrieval.* Supervisor retrieves information from working memory when making decisions. This happens when a fixation or a tapping occurs, and a decision needs to be made about the next action. Supervisor updates its belief about the text that has been typed so far - this includes both the certainty and the correctness. Additionally, it uses the memory of the text already typed to determine the next character.

Figure 3 illustrates how working memory aids in the simulation of human typing. In the concrete case depicted, modeling human working memory can yield a more complete understanding of the situation than simply observing "snapshots" of gaze and finger movements. An ablation study, presented in depth in the supplementary material, supports this conclusion. It revealed that only the model featuring working memory could generate typing performance similar to the human data.

3.3 Training Workflow

We propose a new workflow for training and fitting the computationally rational models. It follows three steps:

(1) Pre-train the components to build the basic human capabilities with the internal environment.
(2) For individual-level differences and generalizability, train the supervisor policy π with randomly sampled cognitive parameters and keyboard screenshots.
(3) For accurate reproduction of data from humans, fit the cognitive parameters $\theta = (E_K, F_K, \lambda)$ of the model to align the simulated behavior with the empirical data.

3.3.1 Pre-training for the internal environment. The three modules are pre-trained separately to work with real-world keyboards. Vision is trained to gaze at the correct position when given a random target (e.g., a key or the text display), Finger is trained to move the finger and tap the correct place for a given random key, and Working Memory is trained to predict what has been typed in light of the information from the pre-trained vision and finger. During training, keyboard screenshots are randomly sampled; Goals, such as looking at the key "h" or pointing to the key "a", are randomly selected; All movements are accompanied by Gaussian-distributed noise. These modules compose the internal environment as the interface between the supervisor and the external environment.

3.3.2 Policy optimization. Step 2 is to train the controller of the model to cover diverse conditions. The output is an optimal policy

π^* that can type well with different cognitive parameters on as many keyboards as possible. Unlike behavioral cloning [16], which involves the expensive step of collecting large quantities of human data, this step does not require *any* human data. The policy is trained through interaction with the pixel-based environment and a designed reward. In this step, we randomly sample target typing phrases from a daily mobile typing dataset [85]. The policy optimization employs two loops, randomly sampling keyboard images and cognitive parameters in the outer loop and learning the optimal policy via reinforcement learning in the inner loop. The reinforcement learning algorithm we used is the proximal policy optimization [82] in stable-baselines3 library [73]. Training the model takes about 6 hours on a commodity GPU computer (NVIDIA GeForce RTX 4090).

3.3.3 Parameter fitting. The trained model can sample diverse users from the cognitive parameter space. The purpose of parameter fitting is to find the optimal parameters $\theta^* = (E_K^*, F_K^*, \lambda^*)$ that can let the typist model perform similarly to the average performance of a target user group. In studies 5.1 and 5.2, we followed this step to fit the parameters to benchmark human data, using Bayesian optimization for efficient parameter-fitting. In each iteration, the typist model samples a set of random trajectories, for comparison with human data via the acquisition function F. This acquisition function measures the similarity between generated behavior and human data:

$$F = \sum_{m \in M} JS(G(m)||H(m)) \tag{4}$$

where M is the list of typing metrics, including typing speed (in WPM), proofreading (the number of gaze shifts), and error correction (the number of backspaces). JS is the Jansen–Shannon divergence, to measure the distance between two distributions (a symmetric and bounded version of the Kullback–Leibler, or KL, divergence). $G(m)$ and $H(m)$, respectively, are the distribution of the performance m derived from the generated and the human data. Bayesian optimization leads to the optimal parameters θ^* of the internal environment that fit the target users. The policy with the optimal parameters $\pi^*(\theta^*)$ can generate typing behavior like the target users'.

4 CREATING A BENCHMARK FOR TOUCHSCREEN-TYPING MODELS

This section presents MobileTyping, a benchmark for evaluating and comparing touchscreen-typing models. The benchmark is released as part of the paper and can be adopted and extended by others. The creation of a benchmark for touchscreen typing is essential for two reasons. First, it allows for evaluation of varied touchscreen-typing models and for their comparison, which is crucial for improving HCI models for typing. Second, it provides a standardized set of data that can be used to train and evaluate machine-learning algorithms for HCI purposes.

4.1 Goals for the Benchmark

We identified three major goals for benchmarking touchscreen typing, informed by prior research in the fields of human–computer interaction and machine learning:

- The touchstone of any simulation model lies in its accuracy, which, in this context, refers to the model's capacity to reliably replicate or forecast human typing actions [18, 47, 51]. An accurate model's policy aligns closely with human strategies, thereby closely mirroring key metrics and phenomena. Taking accuracy as the first priority for modeling human behavior, we sought performance comparable to humans.
- Typing patterns vary considerably among individuals, in line with factors such as finger precision and memory capacity [80]. Some are fast, some slow; some type with more errors, and some proofread more. For example, elderly users may type slowly in response to forgetfulness and declining motor skills [59]. Modeling these individual-level differences can be important, especially for applications that support a special user group [81]. We aimed to generate varied typing behavior that can reflect a wide range of user populations.
- A generalizable computational model is one that should perform well not only for the specific keyboard it was trained on but also for previously unseen keyboards [33]. A model fitted to a specific keyboard might perform well in a lab setting but have a narrower range of real-world usefulness. Previous solutions displayed this limitation: their requirement for manual feature-engineering for new keyboard designs reduced flexibility [43]. The goal of generalizability requires a model that functions well across varied keyboard designs, layouts, and intelligent features.

4.2 Modeling Tasks and Metrics

To measure how well computational models can reach these goals, we developed the benchmark MobileTyping with human typing data and touchscreen keyboards; see Table 1. The table presents all the modeling tasks, arranged into three categories corresponding to the three goals, and comparisons for human ground truth, the latest OSC model [43], and CRTypist. It covers 600 episodes of detail-level gaze and finger movements [41], 18,074 unique users' sentence-level typing performances [66], and 1,028 newly collected keyboard screenshots from a mobile application market. Although MobileTyping covers a large number of participants and designs, this benchmark might still be only a "snapshot" that future work could improve upon. We describe the three categories of modeling tasks in the following subsections.

4.2.1 Accuracy in generating human-like behavior. The first modeling task is to measure the accuracy of prediction. For this, we utilized a typing dataset from researchers' detailed finger-tracking and gaze-tracking with 30 participants who transcribed 20 Finnish-language sentences each in a lab environment [41]. That dataset helped us understand how humans decide on speed, proofreading, and error correction in their typing. We compared the model with human data for single-finger and two-thumb typing both (modeling tasks 1.1 and 1.2 in the table). The following six metrics were chosen for evaluating how accurately the models match human data in terms of typing speed, error correction, and proofreading [4, 23, 88].

- *Word per minute (WPM).* WPM, the most widely used means for assessing typing speed, is computed as the number of standard words (averaging five characters) divided by the time taken [4].

Table 1: MobileTyping: a benchmark for evaluating touchscreen-typing models, including eight modeling tasks with a comparison of models with humans (see details in Sec. 4.2)

Goal	Task	Definition	Metric	Human	OSC [43]	CRTypist
Accuracy in generating human-like behavior	1.1	Reproduce human-like one-finger typing [41]	WPM	27.2 (3.6)	25.2	**28.9** (4.4)
			IKI	381 (51)	399	**366** (30)
			Backspaces	2.6 (1.8)	1.5	**2.4** (2.5)
			Error %	0.6 (0.7)	**0.5**	0.1 (0.4)
			Gaze shifts	3.9 (1.5)	**4.2**	5.5 (1.7)
			Gaze on kbd %	70 (14)	87	**71** (4)
	1.2	Reproduce human-like two-finger typing [41]	WPM	39.3 (10.3)	~32	**34.8** (6.2)
			IKI	267 (64)	376	**275** (19)
			Backspaces	3.6 (2.8)	~0	**5.2** (4.2)
			Error %	0.6 (0.9)	**<0.5**	0.2 (0.8)
			Gaze shifts	3.4 (2.3)	~**3**	4.6 (1.9)
			Gaze on kbd %	60 (16)	-	**73** (4)
Capturing of variety at individuals' level	2.1	Cover differences in one-finger typing [66]	WPM - Max.	70.9	-	53.9
			WPM - Avg.	30.6	-	25.3
			WPM - Median.	29.3	-	21.7
			WPM - Min.	3.9	-	7.6
		Cover differences in two-finger typing [66]	WPM - Max.	96.0	-	64.8
			WPM - Avg.	39.1	-	32.8
			WPM - Median.	37.7	-	25.4
			WPM - Min.	3.4	-	11.4
	2.2	Predict the effects of aging on typing speed [66]	WPM (10-19)	34.1	-	31.0
			WPM (20-29)	32.3	-	29.8
			WPM (30-39)	29.2	-	28.7
			WPM (40-49)	24.4	-	27.1
			WPM (50-59)	22.2	-	25.7
	2.3	Adjust the speed-accuracy tradeoff [66]	WPM	29.2	-	30.6
			Error %	2.3	-	1.4
			Gaze on kbd %	-	-	61
Generalizability of typing ability across diverse keyboard designs, layouts, and features	3.1	Simulate one-finger typing on Gboard [66]	WPM	34.8	-	28.6
			Backspaces	2.4	-	2.5
			Gaze shifts	-	-	4.4
		Simulate one-finger typing on Swiftkey [66]	WPM	32.7	-	28.3
			Backspaces	2.1	-	3.7
			Gaze shifts	-	-	4.8
		Simulate one-finger typing on Go keyboard [66]	WPM	30.5	-	28.4
			Backspaces	2.0	-	3.6
			Gaze shifts	-	-	4.7
	3.2	Simulate one-finger typing on CHUBON keyboard [92]	WPM	33.3	-	34.8
			Backspaces	-	-	3.1
			Gaze shifts	-	-	3.4
		Simulate two-finger typing on KALQ keyboard [65]	WPM	40.2	-	39.2
			Backspaces	-	-	3.2
			Gaze shifts	-	-	2.9
	3.3	Simulate one-finger typing with auto-correct [66]	WPM	31.2	~29	30.9
			Backspaces	2.46	~0.1	3.2
			Gaze shifts	-	~3.7	3.1

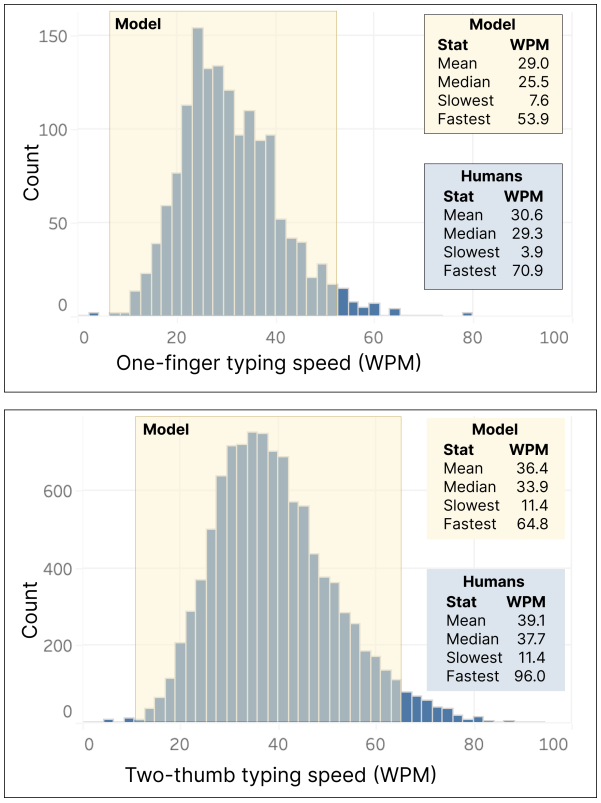

Figure 4: Histograms presenting the distribution of human one- and two-digit typing speed in conditions of no intelligent assistance [66]. The yellow highlighting indicates that CRTypist can cover around 97% of the mass distribution of typing speeds in both forms of typing. Human data typically show a right-skewed distribution; CRTypist yields a similar distribution, with its median performance being slower than the average.

- *Internal-key interval (IKI).* We examined the time, in milliseconds, between consecutive keypresses [23].
- *Number of Backspaces.* Backspacing is a way of removing errors from the typed text. This metric refers to the number of Backspace presses during typing of the given sentence.
- *Error rate (%).* For assessing the correctness of what has been typed, one can compute erroneous characters as a percentage of the total character count.
- *Number of gaze shifts.* Gaze-shifting is movement of the gaze from the keyboard to the text display, which is a signal to proofread the text entered. The amount of gaze-shifting indicates the frequency of proofreading during typing.
- *Gaze-on-keyboard time ratio (%).* The final metric is the percentage of time spent with the gaze on the keyboard. It shows how much time the visual guidance of the finger requires.

4.2.2 Expression of individual differences. The second modeling task assists in measuring representation of individual-specific differences. The dataset for benchmarking here is from large-scale collection of mobile text-entry data from numerous participants performing a web-based transcription task [66]. In Table 1, we provide illustrative statistics for typing speed to summarize the spectrum of individual-level variations. We show peak performance with the max typing speed to see how fast a human typist can reach in each condition. For an overview of performance, we also record the average and median typing speeds. Figure 4 shows the distribution of human single-finger and two-thumb typing speed. One major modeling task (2.1 in Table 1) is to check how much of the mass distribution can be captured. This task setting encourages the computational model to not only replicate the median performance but also cover the whole distribution as much as possible. Alongside this, modeling task 2.2 considers individual differences brought on by age-related changes, and 2.3 entails predicting differences in accordance with the speed–accuracy tradeoff.

4.2.3 Generalizability for diverse visual designs, layouts and features. The modeling tasks connected with the last goal involve diverse typing conditions, with different visual designs (3.1), keyboard layouts (3.2), and auto-correction feature (3.3). In the absence of prior modeling of typing with a wide range of real-world keyboards, we constructed a collection of 1,028 screenshots with real-world touchscreen keyboards. The purpose for this new collection was to build a typing testbed for training and evaluating of computational models over diverse keyboard designs. The collection procedure and details of the results can be found in Supplementary Material. Figure 5 presents a gallery of screenshots with a broad range of visual styles. The richness of the large-scale online collection of data from prior work [66] enables probing, in addition, how humans type on three mainstream keyboards: Gboard, SwiftKey, and GO keyboard. Next, we included two novel keyboards also (CHUBON [92] and KALQ [65]), for predicting single-finger and two-thumb typing with novel layouts. Finally, we added data from human typing with an auto-correct feature [66], to reflect intelligent assistance that is commonplace in daily typing. These task settings (see 3.1 to 3.5 in Table 1) can contribute to assessing how well a model's capturing of typing ability transfers to different conditions.

5 RESULTS

This section presents our evaluation of the model against the tasks defined via MobileTyping (summarized in Table 1).

5.1 Eye and hand movement strategies

Our model demonstrates human-like motion strategies. In particular, it can simulate moment-to-moment eye and finger movements and predict strategies of eye–hand coordination. The simulation results are comparable to empirical typing data (ground truth) [41] and to the output of the latest state-of-the-art OSC approach [43].

5.1.1 Generating human-like single-finger typing. (Task 1.1)

After fitting cognitive parameters (Sec. 3.3.3), CRTypist shows the following human-like behavior as we simulate eye and finger movements (see Fig. 1; Pane a visualizes subject 129 in the the human data; b–c show simulation data; d shows an episode-level comparison on the distribution of internal-key interval (IKI).):

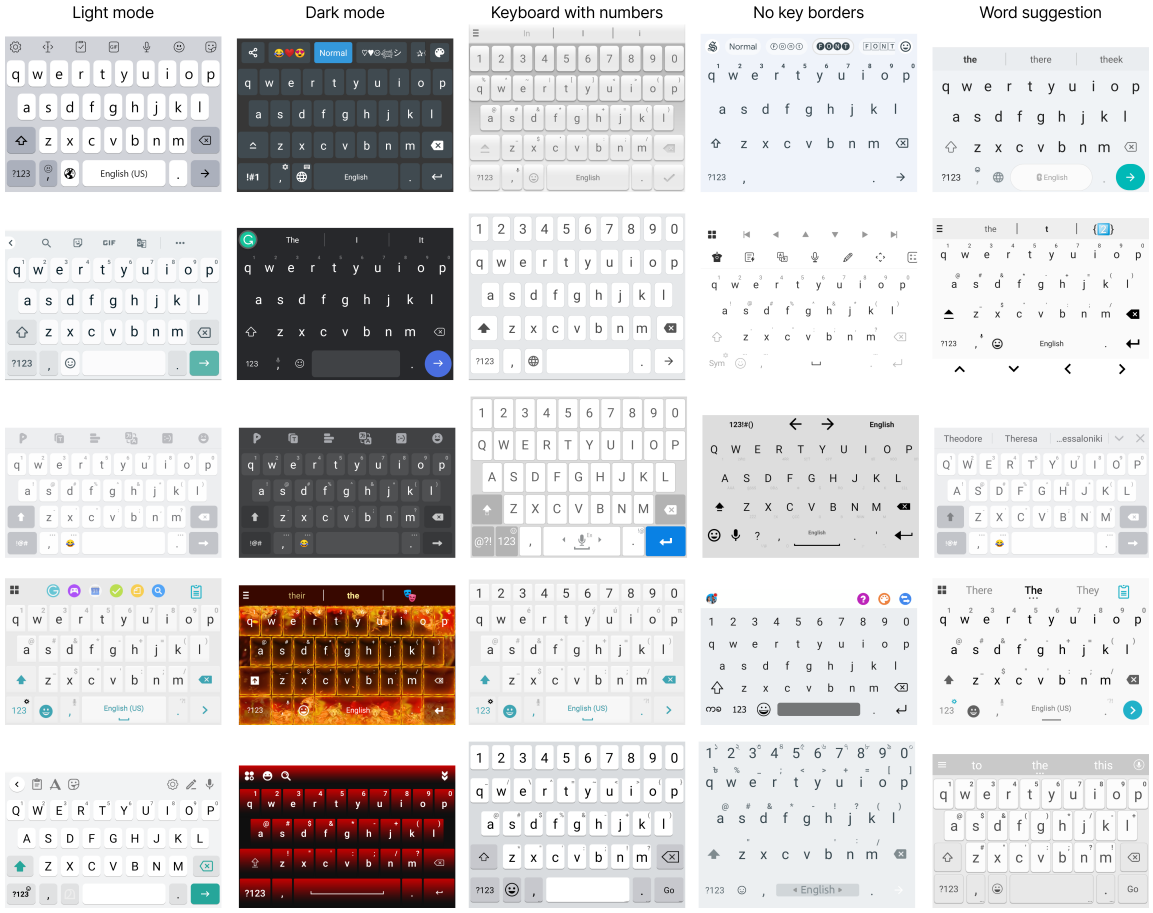

Figure 5: Sample keyboards, representing several categories, from our collection of screenshots in MobileTyping.

1) *Proofreading:* The model has an ability to shift the gaze from the keyboard to the text display to verify what has been written.
2) *Selective focus of gaze:* The model selectively fixates on part of the keyboard instead of all keys tapped.
3) *Error correction:* Correcting typing errors takes place through backspacing.
4) *Parallelism:* Vision and finger get commands concurrently in each timestep, but they do not need to be executed at the same time. The finger can start moving toward the target based on memory of the target position, without fixating on the target. If *vision* is fixating on the target, it can guide the *finger* to land more accurately.
5) *Episode-level similarity:* The predicted finger behavior matches that of humans in terms of typing speed on an episode level.

The model does not simulate full movement trajectories, rather it only simulates the endpoint and the time it takes for the finger or gaze to reach that point, like in earlier work [43]. The interpolation in the figure is solely for a clear visualization. In Figure-c 1, we assume that eye movement is linear [21], while finger movement follows a simple quadratic interpolation that accounts for the time required to home in at the end of the motion [41].

To compare the typing behavior produced by our model to the ground truth and the baseline model, we ran our model with 30 independent episodes (identical to the human data's and the baseline model's conditions). These predictions were aggregated for the comparisons. Our model produces results similar to humans', as shown in Table 1. The WPM, IKI, backspacing, error rate, and gaze-on-keyboard rate fall within one standard deviation of the human data, and the number of gaze shifts falls within two. From comparing the performance of the baseline model and our solution, we conclude that the performance is comparable (see Table 1). Our model outperforms the baseline by the gaze-on-keyboard metric, which the baseline model overestimates relative to the human data.

5.1.2 *Extension to two-thumb typing.* (Task 1.2)

Next, we show how CRTypist performs for two-thumb typing (panel e in Figure 1), which is a popular way of typing on touchscreens [60]. Thanks to its hierarchical and modular design, our model can be easily extended to support two-thumb typing by using two *finger* models, to represent the left and the right thumb. Instead of the constant finger-to-key mapping used in the baseline, we chose a more flexible and realistic implementation: the finger closest to the key is assigned to select it, and a physical constraint for no crossing of fingers is introduced. When a target key is selected,

finger calculates the distance required to move the thumb to tap that key, and then the thumb that requires the shorter movement distance would be selected. Additionally, it ensures that the left thumb is never to the right of the right thumb and vice versa.

Compared to single-finger typing, two-thumb typing is significantly faster (with a 5.9 WPM difference, on average). In addition, IKI falls from 366 to 275 on account of the shorter distances traveled by each finger. Finally, the two-thumb model replicates the phenomenon of fewer gaze shifts relative to single-finger conditions. Table 1 shows the quantitative results and the comparison. The two-thumb typing behavior from CRTypist is comparable to humans', with all metrics falling within one standard deviation from the human data. It performs closer to human data than the baseline approach, especially in Backspace presses.

5.2 Individual differences

Our model can capture users' differing typing capabilities via adjustable cognitive parameters. That is, CRTypist can generate typing behavior that accounts for individuals' differences. This subsection addresses the range of behavior our model can cover, then demonstrates how to simulate typing in various ages and with varying performance objectives.

5.2.1 The model's performance range. (Task 2.1)

The performance range covered by the model can be explored by testing the peak and worst performance. We determine the range of cognitive parameters θ ($E_K \in [0, 0.05]$, $F_K \in [0, 0.18]$, $\lambda \in [0, 0.3]$) by considering empirical data [2, 79, 81]. By setting extreme values to these cognitive parameters, we obtained a maximal and minimal typing speed of 53.9 and 7.6 WPM in one-finger typing and 11.4–64.8 WPM in two-thumb typing. This range indicates that our model can cover 97% of the mass distribution of typing speeds in single-finger and two-thumb typing (see Figure 4). For a better sense of the distribution of the model's performance, we generated 100 independent typing episodes by randomly sampling parameters from the space. The average and median performance in both one- and two-thumb typing show a right-skewed distribution similar to that of the results from human users.

Figure 6 provides a closer look at the peak and worst performance of the model with one target phrase. The former takes less than four seconds, and the latter consumes around 23 seconds. With high cognitive ability (see Figure 6, pane a), the finger moves rapidly with no errors and the vision continues guiding it, with only rare glances at the text display (gaze-on-keyboard ratio: 93%). When set for low cognitive ability (in pane b), the model applies a strategy of proofreading after each keystroke. Finger movement is slow, to assure of accuracy, and the gaze always takes time to check what has been typed (spending 62% of the time on the keyboard).

Our model cannot cover the tails of the two-tailed distribution (see Figure 4), because of the model's underlying assumptions, which do not take into account the extremes. For instance, an expert typist may remember the key positions and expedite typing by means of eyes-free text entry [28]. Our model at present does not cover assumptions of this sort. Likewise, it cannot reproduce the performance of users with special needs, such as some who have Parkinson's disease [86]. This stems from two factors: 1) the error rate for these groups is too high to mesh with our reward function, and 2) the insertion errors they frequently display are not modeled. These issues could be mitigated by revising the reward function to accept a higher error rate and designing a more thorough finger model, with multiple types of errors.

5.2.2 Age-related changes. (Task 2.2)

With another test, we examined individual-specific differences linked to age-related changes. As people grow older, maintenance and processing operations in working memory decline with age [75]. To analyze the effects of decline in working memory on typing speed, we fixed the parameters for *vision* and *finger* and fitted the cognitive parameter λ for working memory to data filtered to summarize typing performance by age band. Specifically, we calculated the average performance (by WPM and backspacing) within each band and fitted the cognitive parameters to this "average user." After parameter fitting to typing speed over different ages, our model can successfully predict the change in typing performance for different age groups (see Table 1). The fitted λ values are 0.0323 (31.0 WPM), 0.0403 (29.8 WPM), 0.1036 (28.7 WPM), 0.2572 (27.1 WPM), and 0.2854 (25.7 WPM). They follow a monotonic trend, thus demonstrating the decline in capacity. A Pearson correlation coefficient was computed to show a linear correlation between typing speed and λ ($r(3) = -0.96$, with $p < 0.01$). The limitation is not considering the change in motor control and vision ability due to aging, which might also have strong correlations with age.

5.2.3 Change in performance objectives. (Task 2.3)

Also, the behavior of touchscreen typists may change as they adjust their performance objectives to the situation at hand. For instance, one may feel more comfortable leaving errors when chatting with friends but type formal letters with greater care. To predict this kind of change in typing behavior, we revised the reward function (Equation 1) and re-train the model. This involves changing the speed–accuracy tradeoff by adjusting weights, which indicates the importance of correctness. In response, when the model typed much faster (speed rose to 30.6 WPM from 28.9 WPM), it made more errors without correction (1.4% rather than 0.1%), and allocated less visual attention to the keyboard (61% instead of 71%).

5.3 Generalizability

CRTypist also has the advantage that its ability can be transferred to unseen keyboards. Below, we evaluate our model with a training and testing set of screenshots, then show how CRTypist adapts to novel keyboard layouts and an auto-correction feature.

5.3.1 Evaluation with real-world touchscreen keyboards. (Task 3.1)

In comparison to preexisting approaches, our model benefits from being able to run simulations for unseen keyboards since it takes the pixels as the input. We tested the transfer capacity of our model with the screenshot collection, splitting it into a training set (28 keyboards, 816 screenshots) and a testing one (10 keyboards, 212 screenshots). CRTypist performs comparably with the training and testing set for typing speed (WPM: $M = 27.6, SD = 4.5$ for training vs. $M = 27.4, SD = 4.4$ for testing), proofreading (gaze shifts: $M = 4.8, SD = 1.8$ vs. $M = 4.6, SD = 1.9$), and error correction (backspaces: $M = 2.9, SD = 2.7$ vs. $M = 3.1, SD = 2.4$). The standard deviation of testing-set error rate, at 1.2%, is slightly higher than the training-set one 0.7%; however, both are in an acceptable

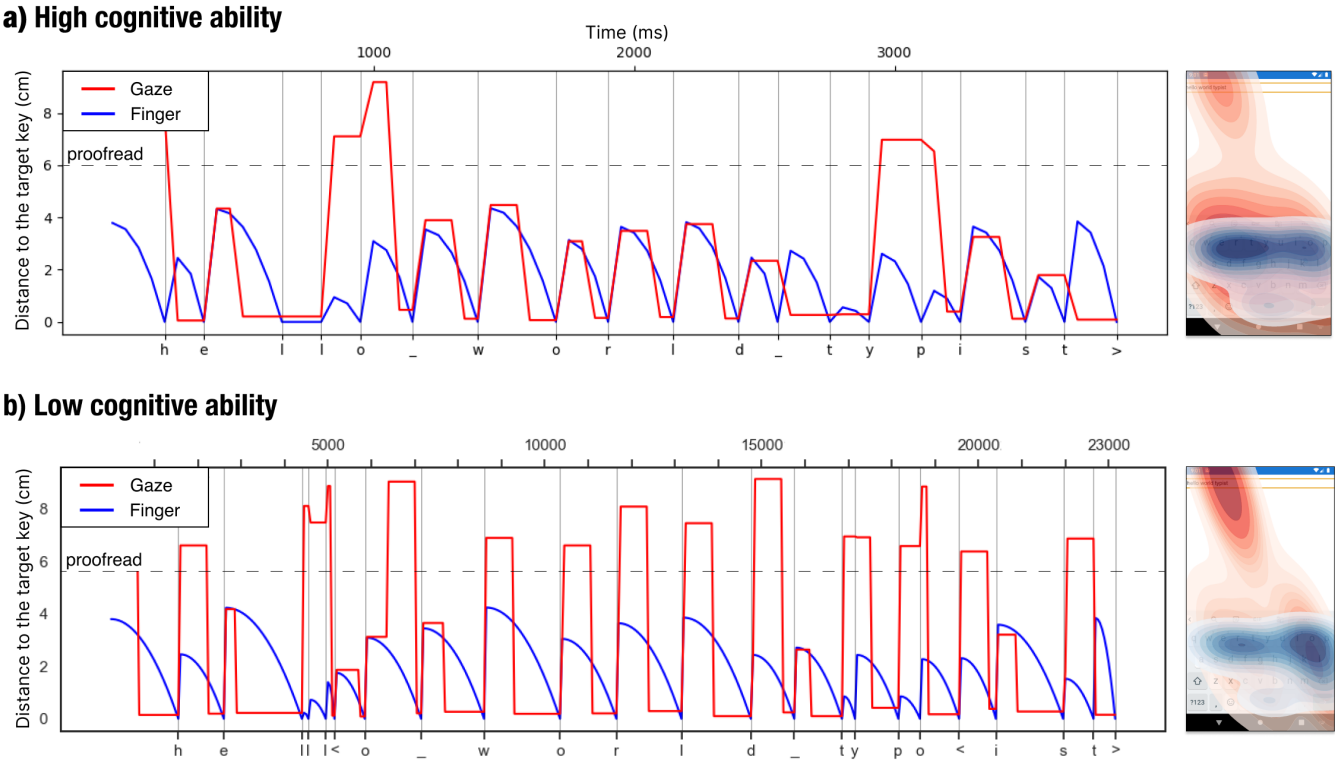

Figure 6: Time-series charts for (a) peak and (b) worst typing behavior. For simulating users with high/low visual, motor, and memory ability, we set all three parameters to the corresponding extreme. The x-axis indicates the time and sequence for the target keys (< and _ denote the backspace and spacebar, respectively) during typing, while the target keys' distance from the gaze and finger is along the y-axis. The space above the dashed line, with gaze distances more than 6 cm from the target, captures looking at the text display for proofreading. a) With high cognitive ability (53.9 WPM), the finger moves quickly in an error-free manner, and the visual system keeps guiding it, with few glances at the text display (the gaze is on the keyboard 93% of the time). b) With low cognitive ability (7.6 WPM): finger motion is slow, for guaranteed accuracy, and the text gets visually checked after each keystroke (gaze-on-keyboard value: 62%). Note that the simulation encompasses only the endpoint and the time for the finger or gaze to reach it; the interpolation in the time-series charts is for clear-visualization purposes only [43].

range of comparability with human performance. Figure 7 (a–c) demonstrates how the trained model types on three mainstream keyboards included in the testing set: Gboard, SwiftKey, and GO Keyboard. No significant performance differences are visible among these three keyboards. Typing is slightly faster with Gboard than on the other keyboards (see Table 1), which could be due to the larger key size in the Gboard screenshot.

5.3.2 Typing on novel-layout keyboards. (Task 3.2)

Next, we tested how the model adapts to two novel layout keyboards: CHUBON [92] and KALQ [65], for one- and two-thumb typing, respectively. The CHUBON layout is optimized specifically for one-finger typing (frequently used letters are near the middle of the keyboard), and KALQ is a layout with proven ability to improve the efficiency of finger use in two-thumb text entry significantly. As CRTypist was trained on QWERTY keyboard screenshots, it cannot be used on non-QWERTY layouts directly. Therefore, we re-trained the internal environment to let CRTypist adapt to these two novel layout keyboards.

The aggregate results are listed in Table 1. In simulation of one-finger typing on CHUBON (Figure 7, pane d), the typing speed, at 34.8 WPM, is much higher than that with QWERTY; this is consistent with the empirical result (33.3 WPM). We can observe from Figure 1 (f) that the finger travels shorter distances than in typing with a QWERTY keyboard; When simulating two-thumb typing on KALQ (Figure 7, pane e), CRTypist predicts nearly equal division of work between the thumbs, and alternating between them is rapid, which suggests frequently switching thumbs while typing. In consequence, the average typing speed on KALQ is 39.2 WPM, which is about 2.6 WPM faster than QWERTY's 36.6 WPM. This speed difference too is in line with that reported from user-study data (4.2 WPM). Note that, because KALQ has two space keys, we removed the space character from the target sentences to address our model's current restriction of one-to-one character–key mappings.

5.3.3 Auto-correction. (Task 3.3)

Research has shown that auto-correction makes typing faster [9], which is an effect our model, when re-trained with this feature,

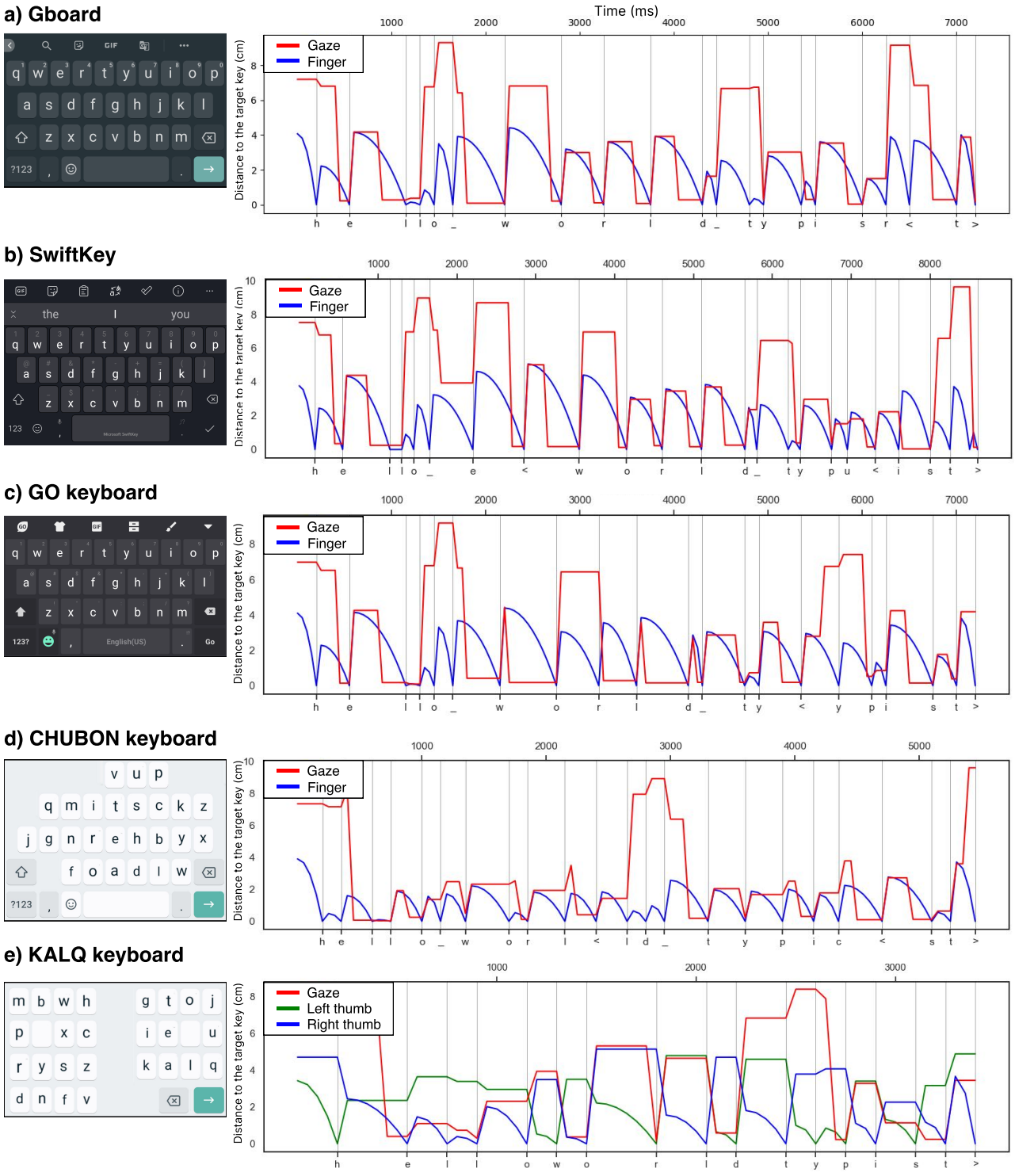

Figure 7: Simulation of single-finger typing with the Gboard (a), SwiftKey (b), GO Keyboard (c), and Chubon keyboard (d) interfaces and with two-thumb typing on the KALQ keyboard (e). In typing on the optimized-layout keyboards (d–e), the finger is fast and the vision spends more time on the keyboard, guiding the fingers.

can reproduce. We used the open-source auto-correction model JamSpell [39], which has a roughly 80% fix rate (i.e., 80% of words with errors end up correct). It corrects the typed sentence once a full word gets followed by a terminating character (a space or Enter). Our model predicts typing 2 WPM faster with this feature than without auto-correction. Additionally, the model relies noticeably on the auto-correcting: it engages in less gaze-shifting and backspacing (see Figure 1, pane g).

6 DISCUSSION

From our study results (see the comparisons in Table. 1), we characterize the findings thus:

- Type like a human: Eye and finger movements generated by CRTypist are comparable to those of human typists in terms of typing speed and strategies for error correction and proofreading.
- Support two-thumb typing: The architecture of CRTypist can be extended to support simulating two-thumb typing. The simulated typing performance is closer to human data than the baseline by four metrics.
- Express diverse users: We observed that CRTypist is able to capture a wide range of individual differences in typing speed, from 7.6 to 64.8 WPM, covering 97% of the mass distribution.
- Predict typing across abilities: By adjusting cognitive parameter λ in working memory, CRTypist shows a strong relationship ($r(3) = -0.96$, with $p < 0.01$) between the typing performance and decay of working memory.
- Reward changes affect objectives: Altering the reward function in CRTypist yields different performance objectives in the speed–accuracy tradeoff (1.7 WPM faster typing speed and 1.3% increase in error rate).
- Generalize to new QWERTY keyboards: CRTypist can generalize to typing on unseen real-world QWERTY keyboards after training. The behavior is comparable with the typing in the training set by all metrics (within one standard deviation); only the standard deviation of error rate shows a more significant difference in relative terms (1.2% and 0.7%).
- Adapt to novel layouts: CRTypist's behavior can adapt to novel-layout keyboards. Its 34.8 WPM on CHUBON and 39.2 WPM on KALQ are consistent with the improvement reported in the original papers.
- Adapt to auto-correction: CRTypist predicts the typing performance improves by about 2 WPM when accurate auto-correction is active. This is in line with prior studies' real-world data.

These results critically build on a key assumption of our model: supervisory control is based on an internal environment (vision, finger, and working memory), which is built over fixations taking place during typing. The internal environment functions as an intermediary between the central controller (supervisor) and the external design (pixels on the touchscreen). This modeling approach allowed us to work directly from pixels without hand-crafted state-action representations. Moreover, separately modeling the underlying cognitive modules opened the door to capturing individual differences by controlling their parameters.

The model offers a significant advancement over existing approaches to modeling touchscreen typing, by simultaneously predicting human-like eye-hand movements, accommodating individual-level factors, and demonstrating generalizability. It successfully produces behavior that adapts to varying conditions tied to individual differences and keyboard designs. These capabilities create the potential for a wide range of applications. Our model could serve as a valuable tool for efficient design evaluation [6], eliminating the need for costly and time-consuming human user tests. For instance, it can facilitate the development of more efficient keyboard layouts to enhance typing speed or evaluate the accessibility of touchscreen keyboards for individuals with disabilities (e.g., those with limited finger accuracy or exhibiting memory impairments), thus promoting accessibility-friendly designs. In addition, the model can simulate typing patterns to develop/refine biometric security measures that exploit keystroke dynamics [38], or simulate player behavior in games that involve text input, thereby enhancing game design and testing [87].

We believe that this model is a leap forward in deploying user models for the text-entry domain. Furthermore, because text entry is plagued by the more general challenge of coordinating limited components (eye–hand coordination), the model's potential extends to simulating user behavior in other interactive tasks, such as visual search [20], pointing [47], and menu selection [19]. Research putting user simulation to such uses holds great promise and, moreover, is essential for researchers striving to understand the behavior of complex systems in HCI [57]. It shows the potential to contribute to the creation and validation of new HCI theories, enhancing the predictability of design and engineering processes, all while improving accessibility. More concretely, simulation-based evaluation can yield immediate insights related to usability before any user testing. Our work represents an exciting prospect for future research into artificial agents that simulate human-like behavior in HCI.

One avenue for future work is to include further capabilities in our model. Inclusion of reading and memorization capabilities will be especially vital for future work. Currently, CRTypist does not model reading behavior [45] in its vision module, an aspect of typing behavior that significantly affects speed. Additionally, the working memory module does not account for long-term memory [61], chunking [91], or the impact of phrase sets. Doing so could provide valuable knowledge of detail-level patterns in human behavior. Our study lays a solid foundation for future research in all these directions. Better modeling of the internal environment should further improve accuracy and facilitate the development of more effective typing interfaces.

A lot of work remains to extend this approach to account for everyday typing more broadly. First, our current model is limited to predicting performance after practice rather than accounting for skill acquisition. As with human users, who can quickly adapt their typing skills to a new QWERTY keyboard, there is also a learning curve when switching to a new keyboard. The process of learning novel layouts poses challenges [44] that require greater attention. Additionally, we need to understand how supervisory control works in actual typing. In this work, we have assumed that humans command finger and gaze movements concurrently. However, this may not accurately reflect how the human cognitive system functions. Moreover, human behavior can be more varied due to the various

features available on a mobile device, such as multitasking (e.g., typing while reading popup notifications), word suggestions [76], and "smart reply" [46]. Further research is required for a comprehensive understanding of these phenomena.

ACKNOWLEDGMENTS

This work was supported by the Research Council of Finland (flagship program: Finnish Center for Artificial Intelligence, FCAI, grants 328400, 345604, 341763; Human Automata, grant 328813; Subjective Functions, grant 357578) and Google Grant (DeepTypist). We thank Prof. Andrew Howes, Dr. Suyog Chandramouli, and all the reviewers for their valuable suggestions and comments.

REFERENCES

[1] John R. Anderson, Daniel Bothell, Michael D. Byrne, Scott Douglass, Christian Lebiere, and Yulin Qin. 2004. An integrated theory of the mind. *Psychological Review* 111, 4 (2004), 1036–1060.

[2] John R. Anderson, Dan Bothell, Christian Lebiere, and Michael Matessa. 1998. An integrated theory of list memory. *Journal of Memory and Language* 38, 4 (1998), 341–380.

[3] Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. 2016. Neural module networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. ACM, 39–48.

[4] Ahmed Sabbir Arif and Wolfgang Stuerzlinger. 2009. Analysis of text entry performance metrics. In *Proceedings of the IEEE Toronto International Conference – Science and Technology for Humanity (TIC-STH 2009)*. IEEE, New York, NY, 100–105.

[5] Shiri Azenkot and Shumin Zhai. 2012. Touch behavior with different postures on soft smartphone keyboards. In *MobileHCI '12: Proceedings of the 14th International Conference on Human–Computer Interaction with Mobile Devices and Services*. ACM, 251–260.

[6] Norman I. Badler, Cary B. Phillips, and Bonnie Lynn Webber. 1993. *Simulating humans: Computer graphics animation and control*. Oxford University Press.

[7] Nikola Banovic, Tovi Grossman, and George Fitzmaurice. 2013. The effect of time-based cost of error in target-directed pointing tasks. In *CHI '13: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 1373–1382.

[8] Nikola Banovic, Varun Rao, Abinaya Saravanan, Anind K. Dey, and Jennifer Mankoff. 2017. Quantifying aversion to costly typing errors in expert mobile text entry. In *CHI '17: Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. ACM, 4229–4241.

[9] Nikola Banovic, Ticha Sethapakdi, Yasasvi Hari, Anind K. Dey, and Jennifer Mankoff. 2019. The limits of expert text entry speed on mobile keyboards with autocorrect. In *MobileHCI '19: Proceedings of the 21st International Conference on Human–Computer Interaction with Mobile Devices and Services*. ACM, Article 15.

[10] Andrew G. Barto and Sridhar Mahadevan. 2003. Recent advances in hierarchical reinforcement learning. *Discrete Event Dynamic Systems* 13, 1–2 (2003), 41–77.

[11] Paul M. Bays, Nikos Gorgoraptis, Natalie Wee, Louise Marshall, and Masud Husain. 2011. Temporal dynamics of encoding, storage, and reallocation of visual working memory. *Journal of Vision* 11, 10, Article 6 (2011).

[12] Xiaojun Bi, Yang Li, and Shumin Zhai. 2013. FFitts law: Modeling finger touch with Fitts' law. In *CHI '13: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, New York, NY, 1363–1372.

[13] Matthew Michael Botvinick. 2012. Hierarchical reinforcement learning and decision making. *Current Opinion in Neurobiology* 22, 6 (2012), 956–962.

[14] Shi Cao, Anson Ho, and Jibo He. 2018. Modeling and predicting mobile phone touchscreen transcription typing using an integrated cognitive architecture. *International Journal of Human–Computer Interaction* 34, 6 (2018), 544–556.

[15] Stuart K. Card. 1983. *The psychology of human–computer interaction*. CRC Press.

[16] Micah Carroll, Rohin Shah, Mark K. Ho, Tom Griffiths, Sanjit Seshia, Pieter Abbeel, and Anca Dragan. 2019. On the utility of learning about humans for human–AI coordination. In *Advances in Neural Information Processing Systems 32 (NeurIPS 2019)*. ACM, 3326–3336.

[17] Kyle R. Cave and Narcisse P. Bichot. 1999. Visuospatial attention: Beyond a spotlight model. *Psychonomic Bulletin & Review* 6 (1999), 204–223.

[18] Xiuli Chen, Aditya Acharya, Antti Oulasvirta, and Andrew Howes. 2021. An adaptive model of gaze-based selection. In *CHI '21: Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. ACM, Article 288.

[19] Xuili Chen, Gilles Bailly, Duncan P. Brumby, Antti Oulasvirta, and Anrew Howes. 2015. The emergence of interactive behavior: A model of rational menu search. In *CHI '15: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, ACM, 4217–4226.

[20] Xiuli Chen, Sandra Dorothee Starke, Chris Baber, and Andrew Howes. 2017. A cognitive model of how people make decisions through interaction with visual displays. In *CHI '17: Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. 1205–1216.

[21] Olivier Coubard, Zoi Kapoula, Rene Muri, and Sophie Rivaud-Péchoux. 2003. Effects of TMS over the right prefrontal cortex on latency of saccades and convergence. *Investigative Ophthalmology & Visual Science* 44, 2 (2003), 600–609.

[22] Andrew T. Duchowski. 2018. Gaze-based interaction: A 30 year retrospective. *Computers & Graphics* 73 (2018), 59–69.

[23] Anna Maria Feit, Daryl Weir, and Antti Oulasvirta. 2016. How we type: Movement strategies and performance in everyday typing. In *CHI '16: Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. ACM, 4262–4273.

[24] Paul M. Fitts. 1954. The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology* 47, 6 (1954), 381–391.

[25] Jerry A. Fodor. 1983. *The modularity of mind*. MIT Press.

[26] Daryl Fougnie, Jordan W. Suchow, and George A. Alvarez. 2012. Variability in the quality of visual working memory. *Nature Communications* 3, 1, Article 1229 (2012).

[27] Michael J. Frank and David Badre. 2012. Mechanisms of hierarchical reinforcement learning in corticostriatal circuits 1: Computational analysis. *Cerebral Cortex* 22, 3 (2012), 509–526.

[28] Dylan Gaines, Mackenzie M. Baker, and Keith Vertanen. 2023. FlexType: Flexible text input with a small set of input gestures. In *IIUI '23: Proceedings of the 28th International Conference on Intelligent User Interfaces*. ACM, 584–594.

[29] Dylan Gaines, John Dudley, Per Ola Kristensson, and Keith Vertanen. 2022. Statistical keyboard decoding. *Bayesian Methods for Interaction and Design* (2022), 188–211.

[30] Julien Gori and Olivier Rioul. 2018. Information-theoretic analysis of the speed–accuracy tradeoff with feedback. In *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, 3452–3457.

[31] Yves Guiard and Olivier Rioul. 2015. A mathematical description of the speed/accuracy trade-off of aimed movement. In *British HCI '15: Proceedings of the 2015 British HCI Conference*. 91–100.

[32] David Gunning, Mark Stefik, Jaesik Choi, Timothy Miller, Simone Stumpf, and Guang-Zhong Yang. 2019. XAI: Explainable artificial intelligence. *Science Robotics* 4, 37, Article eaay7120 (2019).

[33] Trevor Hastie, Robert Tibshirani, and Jerome H. Friedman. 2009. *The elements of statistical learning: Data mining, inference, and prediction, second edition*. Springer.

[34] Geoffrey E. Hinton and Ruslan R. Salakhutdinov. 2006. Reducing the dimensionality of data with neural networks. *Science* 313, 5786 (2006), 504–507.

[35] Eve Hoggan, Stephen A. Brewster, and Jody Johnston. 2008. Investigating the effectiveness of tactile feedback for mobile touchscreens. In *CHI '08: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 1573–1582.

[36] Paul Holleis, Friederike Otto, Heinrich Hussmann, and Albrecht Schmidt. 2007. Keystroke-level model for advanced mobile phone interaction. In *CHI '07: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 1505–1514.

[37] Maija Honig, Wei Ji Ma, and Daryl Fougnie. 2020. Humans incorporate trial-to-trial working memory uncertainty into rewarded decisions. *Proceedings of the National Academy of Sciences* 117, 15 (2020), 8391–8397.

[38] Syed Zulkarnain Syed Idrus, Estelle Cherrier, Christophe Rosenberger, and Patrick Bours. 2014. Soft biometrics for keystroke dynamics: Profiling individuals while typing passwords. *Computers & Security* 45 (2014), 147–155.

[39] JamSpell. 2023. *bakwc/JamSpell: Modern Spell Checking Library*. https://github.com/bakwc/JamSpell

[40] Xinhui Jiang, Jussi P. P. Jokinen, Antti Oulasvirta, and Xiangshi Ren. 2022. Learning to type with mobile keyboards: Findings with a randomized keyboard. *Computers in Human Behavior* 126, Article 106992 (2022).

[41] Xinhui Jiang, Yang Li, Jussi P. P. Jokinen, Viet Ba Hirvola, Antti Oulasvirta, and Xiangshi Ren. 2020. How we type: Eye and finger movement strategies in mobile typing. In *CHI '20: Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. ACM.

[42] Bonnie E. John. 1996. TYPIST: A theory of performance in skilled typing. *Human–Computer Interaction* 11, 4 (1996), 321–355.

[43] Jussi Jokinen, Aditya Acharya, Mohammad Uzair, Xinhui Jiang, and Antti Oulasvirta. 2021. Touchscreen typing as optimal supervisory control. In *CHI '21: Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. ACM, Article 720.

[44] Jussi Jokinen, Sayan Sarcar, Antti Oulasvirta, Chaklam Silpasuwanchai, Zhenxin Wang, and Xiangshi Ren. 2017. Modelling learning of new keyboard layouts. In *CHI '17: Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (Denver, CO). ACM, New York, NY, 4203–4215.

[45] Marcel A. Just and Patricia A. Carpenter. 1980. A theory of reading: From eye fixations to comprehension. *Psychological Review* 87, 4 (1980), 329–354.

[46] Anjuli Kannan, Karol Kurach, Sujith Ravi, Tobias Kaufmann, Andrew Tomkins, Balint Miklos, Greg Corrado, Lśzlø' Lukćs, Marina Ganea, Peter Young, and Vivek Ramavajjala. 2016. Smart reply: Automated response suggestion for email.

In *KDD '16: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 955–964.

[47] Antti Keurulainen, Isak Rafael Westerlund, Oskar Keurulainen, and Andrew Howes. 2023. Amortised experimental design and parameter estimation for user models of pointing. In *CHI '23: Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*. ACM, Article 772.

[48] Davis E. Kieras and Davis E. Meyer. 1997. An overview of the EPIC architecture for cognition and performance with application to human–computer interaction. *Human–Computer Interaction* 12, 4 (1997), 391–438.

[49] Iuliia Kotseruba and John K. Tsotsos. 2020. 40 years of cognitive architectures: Core cognitive abilities and practical applications. *Artificial Intelligence Review* 53, 1 (2020), 17–94.

[50] Benoît Lemaire and Sophie Portrat. 2018. A computational model of working memory integrating time-based decay and interference. *Frontiers in Psychology* 9, Article 416 (2018).

[51] Zhi Li, Yu-Jung Ko, Aini Putkonen, Shirin Feiz, Vikas Ashok, I. V. Ramakrishnan, Antti Oulasvirta, and Xiaojun Bi. 2023. Modeling touch-based menu selection performance of blind users via reinforcement learning. In *CHI '23: Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*. ACM, Article 357.

[52] PyTorch Lightning. 2023. *Char Error Rate – PyTorch-Metrics 1.3.1 Documentation*. https://torchmetrics.readthedocs.io/en/stable/text/char_error_rate.html

[53] Wanyu Liu, Julien Gori, Olivier Rioul, Michel Beaudouin-Lafon, and Yves Guiard. 2020. How relevant is Hick's law for HCI?. In *CHI '20: Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*.

[54] Yaobo Liu, Yeqing Bai, Pengyuan Zhang, Mingyang Zhou, Siwei Wang, Jie Liu, and Shijian Lu. 2021. PaddleOCR: A high performance scene text recognition platform. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. OAE, 2984–2993.

[55] Google LLC. 2023. *Gboard – the Google Keyboard*. https://play.google.com/store/apps/details?id=com.google.android.inputmethod.latin

[56] Hee-Seung Moon, Antti Oulasvirta, and Byungjoo Lee. 2023. Amortized inference with user simulations. In *CHI '23: Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*. ACM, Article 773.

[57] Roderick Murray-Smith, Antti Oulasvirta, Andrew Howes, Jörg Müller, Aleksi Ikkala, Miroslav Bachinski, Arthur Fleig, Florian Fischer, and Markus Klar. 2022. What simulation can do for HCI research. *Interactions* 29, 6 (2022), 48–53.

[58] J. I. Myung and Mark A. Pitt. 1997. Applying Occam's razor in modeling cognition: A Bayesian approach. *Psychonomic Bulletin & Review* 4 (1997), 79–95.

[59] Hugo Nicolau and Joaquim Jorge. 2012. Elderly text-entry performance on touchscreens. In *ASSETS '12: Proceedings of the 14th International ACM SIGACCESS Conference on Computers and Accessibility*. ACM, 127–134.

[60] Hugo Nicolau and Joaquim Jorge. 2012. Touch typing using thumbs: Understanding the effect of mobility and hand posture. In *CHI '12: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, New York, NY, 2683–2686.

[61] Dennis Norris. 2017. Short-term memory and long-term memory are still different. *Psychological Bulletin* 143, 9 (2017), 992–1009.

[62] Antti Oulasvirta and Kasper Hornbæk. 2022. Counterfactual thinking: What theories do in design. *International Journal of Human–Computer Interaction* 38, 1 (2022), 78–92.

[63] Antti Oulasvirta, Jussi P. P. Jokinen, and Andrew Howes. 2022. Computational rationality as a theory of interaction. In *CHI '22: Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*. ACM, Article 359.

[64] Antti Oulasvirta, Sunjun Kim, and Byungjoo Lee. 2018. Neuromechanics of a button press. In *CHI '18: Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. ACM, Article 508.

[65] Antti Oulasvirta, Anna Reichel, Wenbin Li, Yan Zhang, Myroslav Bachynskyi, Keith Vertanen, and Per Ola Kristensson. 2013. Improving two-thumb text entry on touchscreen devices. In *CHI '13: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2765–2774.

[66] Kseniia Palin, Anna Maria Feit, Sunjun Kim, Per Ola Kristensson, and Antti Oulasvirta. 2019. How do people type on mobile devices? Observations from a study with 37,000 volunteers. In *MobileHCI '19: Proceedings of the 21st International Conference on Human–Computer Interaction with Mobile Devices and Services*. ACM, Article 9.

[67] Pekka Parhi, Amy K. Karlson, and Benjamin B. Bederson. 2006. Target size study for one-handed thumb use on small touchscreen devices. In *MobileHCI '06: Proceedings of the 8th Conference on Human–Computer Interaction with Mobile Devices and Services*. ACM, 203–210.

[68] David Lorge Parnas. 1972. On the criteria to be used in decomposing systems into modules. *Commun. ACM* 15, 12 (1972), 1053–1058.

[69] Benjamin Pearson, Julius Raškevičius, Paul M. Bays, Yoni Pertzov, and Masud Husain. 2014. Working memory retrieval as a decision process. *Journal of Vision* 14, 2, Article 2 (2014).

[70] Yoni Pertzov, Paul M. Bays, Sabine Joseph, and Masud Husain. 2013. Rapid forgetting prevented by retrospective attention cues. *Journal of Experimental Psychology: Human Perception and Performance* 39, 5 (2013), 1224–1231.

[71] Richard W. Pew. 1966. Acquisition of hierarchical control over the temporal organization of a skill. *Journal of Experimental Psychology* 71, 5 (1966), 764–771.

[72] Sophie Portrat, Pierre Barrouillet, and Valérie Camos. 2008. Time-related decay or interference-based forgetting in working memory? *Journal of Experimental Psychology: Learning, Memory, and Cognition* 34, 6 (2008), 1561–1564.

[73] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. 2021. Stable-Baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research* 22, 268 (2021).

[74] Keith Rayner. 2009. The 35th Sir Frederick Bartlett Lecture: Eye movements and attention in reading, scene perception, and visual search. *Quarterly Journal of Experimental Psychology* 62, 8 (2009), 1457–1506.

[75] Patricia A. Reuter-Lorenz and Ching-Yune C. Sylvester. 2005. The cognitive neuroscience of working memory and aging. In *Cognitive Neuroscience of Aging: Linking Cognitive and Cerebral Aging*, R. Cabeza, L. Nyberg, and D. Park (Eds.). Oxford University Press, 186–217.

[76] Quentin Roy, Sébastien Berlioux, Géry Casiez, and Daniel Vogel. 2021. Typing efficiency and suggestion accuracy influence the benefits and adoption of word suggestions. In *CHI '21: Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. ACM, Article 714.

[77] David E. Rumelhart and Donald A. Norman. 1982. Simulating a skilled typist: A study of skilled cognitive-motor performance. *Cognitive Science* 6, 1 (1982).

[78] Timothy A. Salthouse. 1986. Perceptual, cognitive, and motoric aspects of transcription typing. *Psychological Bulletin* 99, 3 (1986), 303–319.

[79] Dario D. Salvucci. 2001. An integrated model of eye movements and visual encoding. *Cognitive Systems Research* 1, 4 (2001), 201–220.

[80] Sayan Sarcar, Jussi P. P. Jokinen, Antti Oulasvirta, Zhenxin Wang, Chaklam Silpasuwanchai, and Xiangshi Ren. 2018. Ability-based optimization of touchscreen interactions. *IEEE Pervasive Computing* 17, 1 (2018), 15–26.

[81] Sayan Sarcar, Jussi Joklnen, Antti Oulasvirta, Chaklam Silpasuwanchai, Zhenxin Wang, and Xiangshi Ren. 2016. Towards ability-based optimization for aging users. In *ITAP '16: Proceedings of the International Symposium on Interactive Technology and Ageing Populations*. ACM, 77–86.

[82] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).

[83] Matthijs T. J. Spaan. 2012. Partially observable Markov decision processes. In *Reinforcement Learning: State of the Art*, Marco Wiering and Martijn van Otterlo (Eds.). Springer, 387–414.

[84] Apple Support. 2023. *Type with the Onscreen Keyboard on iPhone*. https://support.apple.com/guide/iphone/type-with-the-onscreen-keyboard-iph3c50f96e/ios

[85] Keith Vertanen and Per Ola Kristensson. 2011. A versatile dataset for text entry evaluations based on genuine mobile emails. In *MobileHCI '11: Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services*. ACM, 295–298.

[86] Yuntao Wang, Ao Yu, Xin Yi, Yuanwei Zhang, Ishan Chatterjee, Shwetak Patel, and Yuanchun Shi. 2021. Facilitating text entry on smartphones with QWERTY keyboard for users with Parkinson's disease. In *CHI '21: Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. ACM, Article 735.

[87] Christoph Wimmer, Bernd Stainer, and Thomas Grechenig. 2022. On the impact of competitive gameplay on text entry performance – a study based on a mobile typing game. In *CHI EA '22: Extended Abstracts of the 2022 CHI Conference on Human Factors in Computing Systems*. ACM, Article 430.

[88] Jacob O. Wobbrock. 2007. Measures of text entry performance. *Text Entry Systems: Mobility, Accessibility, Universality* (2007), 47–74.

[89] Jeremy M. Wolfe, Kyle R. Cave, and Susan L. Franzel. 1989. Guided search: An alternative to the feature integration model for visual search. *Journal of Experimental Psychology: Human Perception and Performance* 15, 3 (1989), 419–433.

[90] Yanfei Xie, Grace P. Y. Szeto, Jie Dai, and Pascal Madeleine. 2016. A comparison of muscle activity in using touchscreen smartphone among young people with and without chronic neck–shoulder pain. *Ergonomics* 59, 1 (2016), 61–72.

[91] Motonori Yamaguchi and Gordon D. Logan. 2014. Pushing typists back on the learning curve: Revealing chunking in skilled typewriting. *Journal of Experimental Psychology: Human Perception and Performance* 40, 6 (2014), 1713–1732.

[92] Shumin Zhai, Michael Hunter, and Barton A. Smith. 2000. The Metropolis keyboard – an exploration of quantitative techniques for virtual keyboard design. In *UIST '00: Proceedings of the 13th Annual ACM Symposium on User Interface Software and Technology*. ACM, 119–128.