I-Keyboard: Fully Imaginary Keyboard on Touch Devices Empowered by Deep Neural Decoder

Ue-Hwan Kim, Sahng-Min Yoo, and Jong-Hwan Kim, *Fellow, IEEE*

***Abstract*—Text entry aims to provide an effective and efficient pathway for humans to deliver their messages to computers. With the advent of mobile computing, the recent focus of text-entry research has moved from physical keyboards to soft keyboards. Current soft keyboards, however, increase the typo rate due to a lack of tactile feedback and degrade the usability of mobile devices due to their large portion on screens. To tackle these limitations, we propose a fully imaginary keyboard (I-Keyboard) with a deep neural decoder (DND). The invisibility of I-Keyboard maximizes the usability of mobile devices and DND empowered by a deep neural architecture allows users to start typing from any position on the touch screens at any angle. To the best of our knowledge, the eyes-free ten-finger typing scenario of I-Keyboard which does not necessitate both a calibration step and a predefined region for typing is first explored in this article. For the purpose of training DND, we collected the largest user data in the process of developing I-Keyboard. We verified the performance of the proposed I-Keyboard and DND by conducting a series of comprehensive simulations and experiments under various conditions. I-Keyboard showed 18.95% and 4.06% increases in typing speed (45.57 words per minute) and accuracy (95.84%), respectively, over the baseline.**

***Index Terms*—Decoding, eyes-free, human–computer interaction (HCI), imaginary keyboard (I-Keyboard), soft keyboard, text entry, user experience, user interfaces (UIs), virtual keyboard.**

I. Introduction

TEXT-ENTRY takes a crucial role in human–computer interaction (HCI) applications [1]. It is one of the most effective and efficient methods for humans to deliver messages to computers. In the early stage of HCI research, physical keyboard-based text-entry methods were prevailing. Since then, researchers have focused on designing keyboards with high usability. In the post-PC era, the birth of mobile [2] and ubiquitous computing has prompted the development of soft keyboards. Soft keyboards [3] set mobile devices free from equipping additional hardware, thus improving the mobility of mobile devices.

However, contemporary soft keyboards possess a few limitations. In fact, current soft keyboard techniques damage the usability of mobile devices in multiple ways other than the mobility. First, the lack of tactile feedback increases the rate of typos. The absence of tactile feedback causes hand drifts and tap variability among people when typing with soft keyboards [4]. The users gradually misalign their hands and touch different points for the same input without the physical boundaries. In addition, users type in an eyes-free manner [5] in a couple of mobile computing contexts where a display is separated from the typing interface. In such situations, the effect of hand drift becomes greater.

Second, soft keyboards hinder mobile devices from presenting enough content because they occupy a relatively large portion on displays. Mobile devices provide smaller displays than nonmobile devices in general and soft keyboards can fill up to 40% of displays. A survey of over 50 iPad users [6] indicates that users show dissatisfaction toward current soft keyboard dimensions. The users wish to minimize the size of the virtual keyboard to view more information from the display. In short, we claim that current soft keyboards on mobile devices degrade the usability of mobile screens.

In this article, we propose the imaginary keyboard (I-Keyboard) along with the deep-learning (DL)-based decoding algorithm to tackle the above-mentioned limitations of soft keyboards. First, the proposed I-Keyboard is invisible, which maximizes the utility of the screens on mobile devices. The users can view the content of an application in full screen and type freely at the same time. To further improve usability, I-Keyboard does not have a predefined layout, shape, or size of keys. The users can start typing on any position at any angle on touch screens without worrying about the keyboard position and shape.

Next, I-Keyboard comes with a DL-based decoding algorithm which does not require a calibration step. The proposed deep neural decoder (DND) effectively handles both hand drift and tap variability and dynamically translates the touch points into words. In order to design DND for I-Keyboard, we conduct a user study to analyze user behaviors and collect data to train DND. To the best of our knowledge, this data is the largest dataset for designing a soft keyboard decoder. For training DND, we formulate an auxiliary loss without which deep architectures do not converge. Moreover, the eyes-free ten-finger typing scenario which not only discards a calibration

Manuscript received July 9, 2019; revised October 7, 2019; accepted November 2, 2019. This work was supported in part by the Institute for Information and Communications Technology Promotion grant funded by the Korea Government (MSIT) Research on Adaptive Machine Learning Technology Development for Intelligent Autonomous Digital Companion under Grant 2016-0-00563, and in part by the National Research Foundation of Korea (NRF) grant funded by the Korea Government (MSIT) under Grant NRF-2017R1A2A1A17069837. This article was recommended by Associate Editor H. A. Abbass. *(Corresponding author: Jong-Hwan Kim.)*

The authors are with the School of Electrical Engineering, Korea Advanced Institute of Science and Technology, Daejeon 34141, South Korea (e-mail: uhkim@rit.kaist.ac.kr; smyoo@rit.kaist.ac.kr; johkim@rit.kaist.ac.kr).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCYB.2019.2952391

step but also liberates users from typing at a predefined area has not been explored. The proposed I-Keyboard attempts to unfold such a truly imaginary soft keyboard for the first time.

In summary, the main contributions of this article are as follows.

1) *Research Scenario:* We define an advanced typing scenario where both a predefined typing area and a calibration step are omitted.
2) *Data Collection:* We collect user data in an unconstrained environment and comprehensively analyze user behaviors in such an environment.
3) *DND:* We design a DL architecture for a shape-, position-, and angle-independent decoding and formulate the auxiliary loss for training DND.
4) *Verification:* We verify the effectiveness of the proposed I-Keyboard through extensive simulations and experiments.
5) *Open Source:* We make the materials developed in this article public[1]: the source code for the data collection tool, the collected data, the data preprocessing tool, and the I-Keyboard framework.

The remainder of this article is structured as follows. In Section II, we review conventional research approaches and compare them with the proposed I-Keyboard. Section III describes the data collection process and analyzes user behaviors. Section IV delineates the I-Keyboard system architecture and DND. Sections V and VI verify the performance of I-Keyboard through comprehensive simulations and experiments. Discussion points follow in Section VII and concluding remarks in Section VIII.

II. Related Work

In this section, we review previous research outcomes relevant to the proposed I-Keyboard. We discuss the main ideas and limitations of the previous works and compare them with the proposed text-entry system.

A. Intelligent Text Entry

Intelligent text entry aims to provide quick and accurate typing interfaces to users. Although mechanical keyboards can also employ intelligent text-entry schemes, the schemes in general target virtual keyboards since virtual keyboards offer simpler ways to modify the keyboards. In this article, we categorize the intelligent text-entry schemes into three classes: 1) gesture based; 2) optimized; and 3) imaginary text entries.

First, gesture-based text entry allows drawing-like typing [7], [8]. Drawing-like typing removes the need for localizing each key position and users can start drawing from any place on the screen in an eyes-free manner. Though gesture-based text entry offers concise eyes-free typing interfaces, it requires gesture recognition algorithms, which can hardly achieve a high accuracy [9]. Gesture variability among users and similarities between gestures for each key cause ambiguity that increases the inherent difficulty of sequence classification [10]. In addition, gesture-based text entry takes longer time than other text-entry methods since each key involves a gesture rather than a touch or a key press. The proposed I-Keyboard targets a more tractable decoding problem and utilizes the DL techniques to successfully deal with the variability among users.

Second, optimized text entry supplies accessible and comfortable typing interfaces by optimizing the size, shape, and position of keys [11], [12]. Current optimized text-entry methods require users to learn new typing interfaces [13] because knowledge transfer seldom occurs for novel typing interfaces. Furthermore, the optimization process frequently demands a calibration step. The calibration step complicates the usage of the optimized text entries. I-Keyboard proposed in this article does not involve learning and calibration processes since it operates with ten fingers and its decoding algorithm does not need any prior knowledge.

Last but not least, imaginary keyboards, which are invisible to users, save invaluable screen resources and enables multitasking in the context of mobile computing [14]. Imaginary keyboards reduce constraints during interaction and users can freely and comfortably deliver their messages. In addition, the imaginary keyboards coincide with the potent vision for user interfaces (UIs) which has evolved from mechanical, graphical, and gestural UI to imaginary UI, achieving tighter embodiment and more directness [15]. Conventional works on imaginary UI, however, have only shown the feasibility not reaching the practical deployment level. Our I-Keyboard proposes a new concrete concept for imaginary UI and demonstrates the practical implementation of the concept deployed in a real-world environment.

B. Ten-Finger Typing

Ten-finger typing is one of the most natural and common text-entry methods [16]. The users can achieve a typing speed of 60–100 words per minute (WPM) by ten-finger typing on physical keyboards [17]. Ten-finger typing experience stored in muscle memory and tactile feedback from mechanical keys enable eyes-free typing [18]. However, transferring ten-finger typing knowledge from mechanical keyboards to soft keyboards does not successfully occur in general due to lack of tactile feedback, though a few works have shown the viability in special use cases [19], [20].

A number of works have attempted to understand the user behavior with ten-finger typing with soft keyboards. The major findings are as follows.

1) *Speed:* The typing speed with soft keyboards drops dramatically compared to the speed with physical keyboards [21].
2) *Typing Pattern:* The distribution of touch points resembles the mechanical keyboard layout [22], though the distribution varies over time in shape and size [17].
3) *Hand Drift:* Hand drift occurs over time and becomes stronger for invisible keyboards [4].
4) *Tap Variability:* Various factors cause tap variability among users [23]. The factors include finger volume, hand posture, and mobility.

[1] https://github.com/Uehwan/I-Keyboard

In summary, ten-finger eyes-free typing on virtual keyboards, which is most natural and easy to transfer knowledge directly from physical keyboards [17], is feasible according to the previous research results, though a couple of obstacles need to be resolved. The proposed DND handles hand drift, tab variability, and automatic calibration with a deep neural architecture to improve the typing speed and to reduce the error rate.

C. Decoding Algorithms

Classical statistical decoding algorithms translate user inputs (keystrokes) into characters or words using the probabilistic models. These statistical decoding algorithms have proved their effectiveness in a few controlled environments [24]. The goal of statistical decoding is to find the sequence of characters that maximizes the joint probability of the given user input sequence. Mathematically, the user typing pattern for the decoding process is formulated as

$$\hat{\mathbb{C}} = \arg\max_{\mathbb{C}} P(t_1, \ldots, t_n | c_1, \ldots, c_n) \tag{1}$$

where t_is are the position of each keystroke, and $\hat{\mathbb{C}} = (\hat{c}_1, \ldots, \hat{c}_n)$, c_is are the characters and n is the length of the sequence. Since the complexity of modeling the joint probability becomes untractable as the sequence length increases, the independence assumption is employed in most cases. By assuming the independence property, (1) becomes

$$\hat{\mathbb{C}} \simeq \arg\max_{\mathbb{C}} \prod_{i=1}^{n} P(t_i | c_i). \tag{2}$$

The probability $P(t_i|c_i)$ is approximated by a Gaussian distribution with the Markov–Bayesian algorithm [17] or a bivariate Gaussian distribution [25] in conventional approaches. In addition, the probability is separately modeled for left and right hands.

The conventional statistical decoding algorithms, however, cannot perfectly deal with the complex dynamics of user inputs. The independence assumption applied in these methods cannot count both long-term and short-term dependencies among the keystrokes. The independence assumption confines the conventional approaches to regard only the current input. Furthermore, the previous research outcomes have proposed the fixed models for statistical decoding, thus they cannot adaptively handle hand drift and tap variability that vary over time. Though a few have designed adaptive models, those models require either an additional calibration step or a controlled experiment environment [17].

On the other hand, the proposed I-Keyboard comes with DND that overcomes the limitations of the previous approaches. We train DND to translate the user input data containing hand drifts and tap variability into a sequence of characters. The deep neural architecture for DND inherently models the long-term dependency without relying on the independence assumption and its implicit dynamic feature successfully eliminates the effect of hand drift and tap variability. Moreover, the semantic embedding integrated in the neural architecture functions as a language model that boosts the decoding performance. The reduced computational cost due to the joint architecture of decoding and language models is another advantage.

Fig. 1. Data collection environment. We set up an ordinary monitor and a touch screen on the desk.

III. User Study: Understanding User Behavior

In order to understand the user behaviors on invisible and layout-free keyboards, we design a user study. Specifically, we aim to answer three questions: 1) can users type on touch screens without any visual and physical guidances from the screens? 2) if they could, what are the characteristics of their typing behaviors in such an environment? and 3) what type of feedback could improve the user experience during data collection?

A. Data Collection

1) Participants: We recruited 43 participants from the campus (32 males and 11 females) aged from 22 to 32 (average = 25.84, std = 2.68). All participants regularly use both physical QWERTY keyboards and soft keyboards on touchable mobile devices. All of them could type in an eyes-free manner with physical keyboards.

2) Apparatus: Fig. 1 shows the overall data collection environment. We set up one touch screen (LG 23ET63, 1920×1080 pixels, 555×338 mm^2, touchable) for text entry and one ordinary screen (LG L1980QY, 1280×1024 pixels, 422×410 mm^2, nontouchable) for instruction on a desk in a room for data collection. We placed the touch screen parallel to the desk so that users could treat the screen as a usual keyboard. We placed the ordinary screen for instruction perpendicular to the desk.

3) Typing Interface: Fig. 2 illustrates the interface of the instruction and the typing screens. We implemented the interface using windows presentation foundation (WPF). The instruction screen shows the task sentence (phrase) at the center. During typing, three types of feedback could be offered: none, asterisk [17], [25], and highlight. For highlight feedback which is evaluated in this article for the first time, each character of the task sentence gets background-highlighted step by step as a legal touch is detected. The highlight feedback ensures one-to-one mapping between the touch points and the characters in the task sentence.

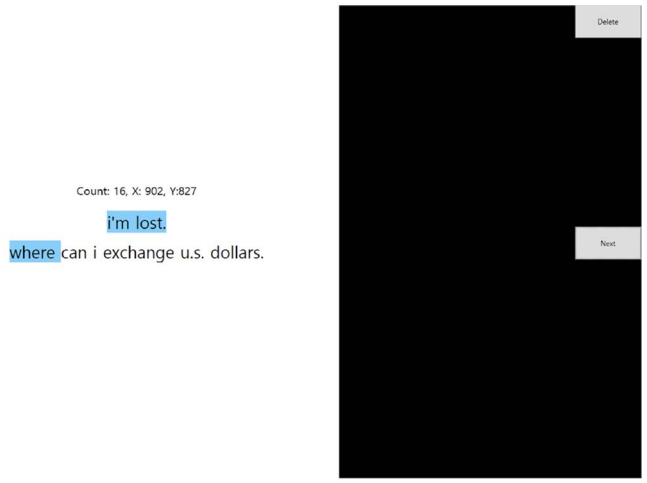

Fig. 2. Typing interface. Only the center part of the instruction screen and the right half part of the touch screen are shown. Interface of the (a) instruction screen and (b) touch screen.

Next, users could start typing from anywhere on the typing screen at any angle. Neither a predetermined area nor the constraints were set for typing. The typing screen only contains the proceed and the delete buttons. Completing the current task sentence activates the proceed button and users can touch no more than the length of the task sentence. The users can delete the touch points collected for the current sentence in case users feel they have made typing mistakes.

4) Procedure: We first collected the demographics of participants and instructed the participants about the data collection procedure. We asked the users to assume that there exists a keyboard on the touch screen and type as naturally as possible. For the first 15 sentences, participants familiarized themselves with the typing interface. After the warm-up, users transcribed 150–160 sentences randomly sampled from the Twitter and 20 Newsgroup datasets. We preprocessed the task sentences so that the sentences only included 26 English letters in lowercase, enter, space, period (.), and apostrophe (’). We randomly connected two sentences with the enter key. The data collection process for each participant took approximately 50 min. In total, the data collection included 7245 phrases and 196 194 keystroke instances. Table I compares the dataset size collected in this article with those of other studies and verifies that the dataset is the largest among the recent studies.

In addition, we randomly selected nine of the participants and asked them to evaluate the three types of feedback. The selected participants typed ten phrases with each feedback, totalling 30 phrases. After the typing session, we collected three subjective ratings from the participants for each feedback type in 5-Likert scale: intuitiveness, convenience, and overall satisfaction.

B. User Behavior Analysis

1) User Mental Model: Fig. 3 exemplifies the user mental models for the keyboard. We normalized the scales and removed the location offsets. Each user mental model characterizes different shape, size, and location. In contrast to the previous reports where the users have typed on the predefined regions, the keys do not align on straight lines, but rather on curves. We presume this results from the fact that the participants in this article have typed in a less-restricted environment. The participants in this article could naturally exhibit their typing behaviors with more freedom.

Though each user recognizes the keyboard layout in mentally different ways, each model consistently resembles the physical keyboard layout. This indicates that users can type on touch screens even though there is no visual and physical guidances.

2) Physical Dimensions: To estimate the physical dimensions, we utilized the distance between the space and the character “p” in each sentence. We omitted the sentences that did not include “p” for the analysis. Table II summarizes the analysis results. First, the average scale ratios among participants are 0.96 and 1.00 for horizontal and vertical directions,

TABLE I
COMPARISON OF DATASET SIZE

	Participants	Age	Phrases	Points
BlindType [23]	12	$20-32$	$9,664$ (words)	$40,509$
TOAST [17]	15	22 (mean)	450	$12,669$
Invisible Keyboard [25]	18	$18-50$	$2,160$	N.A.
Ours	43	25 (mean)	$7,245$	$196,194$

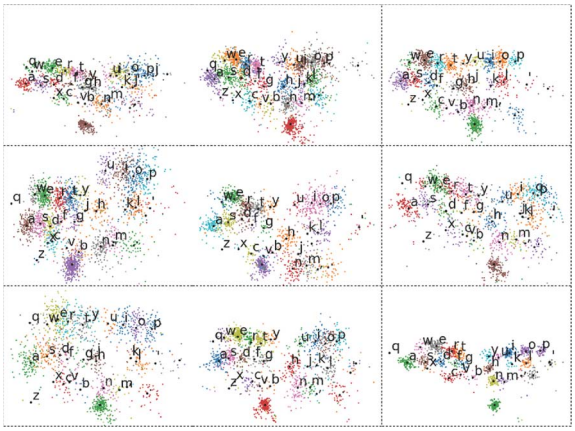

Fig. 3. Examples of the user mental models. The scales are normalized and the offsets are removed for visualization.

TABLE II
SUMMARY OF USER BEHAVIOR ANALYSIS

Metric	Direction	Average	S.D.	Range
Scale	Horizontal	0.96	0.16	$(0.92, 1.02)$
	Vertical	1.00	0.13	$(0.96, 1.04)$
Offset	Horizontal	44.73	75.81	$[0.00, 70.07)$
	Vertical	25.95	44.69	$[0.00, 45.57)$
Size	Horizontal	259.0	87.57	$(246.5, 276.2)$
	Vertical	125.9	22.11	$(120.4, 130.4)$

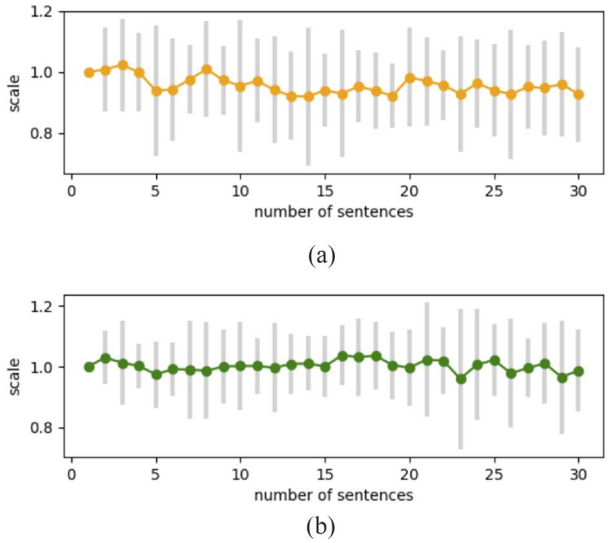

Fig. 4. Variation of scale over time. Scale does not change much in both horizontal and vertical directions indicating consistent user mental model size over time. Variation of the scale in the (a) horizontal direction and (b) vertical direction.

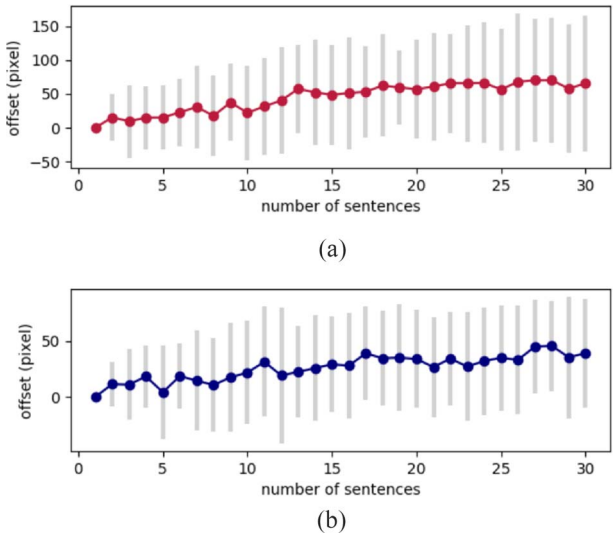

Fig. 5. Variation of offset over time. Offset increases in both horizontal and vertical directions indicating user mental model drifts over time. Variation of the offset in the (a) horizontal direction and (b) vertical direction.

respectively. On average, users keep the overall dimension of the mental keyboard. The low standard deviations of the scales imply that the scales do not alter significantly over time. Fig. 4 further examines the variation of the mental keyboard scales in both directions over time. As implied by the standard deviation, the scales do not vary greatly in both directions.

Next, the average offsets for horizontal and vertical directions are 44.73 and 25.95 pixels, respectively, whereas the standard deviations are 75.81 and 44.69 for each direction, respectively. We calculated offsets compared with the mental keyboard model estimated from the first typed sentence. The users tend to move in both horizontal and vertical directions, though the tendency is stronger in the horizontal direction. Fig. 5 investigates the change of the mental keyboard offsets over time. The mental models drift away from the initial position in both directions as expected.

Finally, the average size of the user mental models is smaller than the size of an actual physical keyboard (350×150 mm^2) in both directions. This contradicts the previous report which has stated that user mental models are slightly larger than an actual keyboard [17]. We surmise that the difference results from the different instructions provided to users and the different data collection settings. Compared with the previous work, we did not enforce any restrictions on the typing environment, such as a predefined typing area or an initialization process for calibration. Thus, we expect this article can examine user behavior in a more natural and precise manner. We do not plot the variation of the size over time since it shows the same tendency as the scales with a different dimension (scale factor).

3) Typing Feedback: Table III shows the user experience results depending on the feedback types. The results demonstrate that highlight feedback used in this article is more intuitive, convenient, and satisfactory than the conventional asterisk feedback. When no feedback is given, users could hardly type the given text during the data collection process.

TABLE III
COMPARISON OF THREE TYPES OF FEEDBACK

Metric	None	Asterisk	Highlight
Intuitiveness	1.13	3.63	4.63
Convenience	1.5	3.5	4.25
Overall Satisfaction	1.25	3.25	4.13

IV. I-KEYBOARD SYSTEM DESIGN

We describe the overall architecture of the proposed I-Keyboard and DND in this section. DND proposed in this article allows the efficient and effective translation of user inputs into character sequences.

A. System Architecture

Fig. 6 describes the overall system architecture of I-Keyboard. The user interaction module receives the user input through a touch interface and displays the decoded sequence through a display interface. The touch interface and the display interface can be contained in one device, though the present work separates the two. The data preparation module preprocesses and formats the raw user input and stores the data in a user input buffer. Since the proposed DND can consider both short-term and long-term dependencies present in the user data, the data from the past time steps improve the decoding performance. For this, the user input buffer keeps a certain number of past input data as a context. After the number of past input data gets stored in the user input buffer, the decoding becomes instant. The new input character is decoded with the past inputs stored in the user input buffer.

The communication layer between the data module and the DND module enables a tight integration of a DL framework and an application framework. Technically, current DL

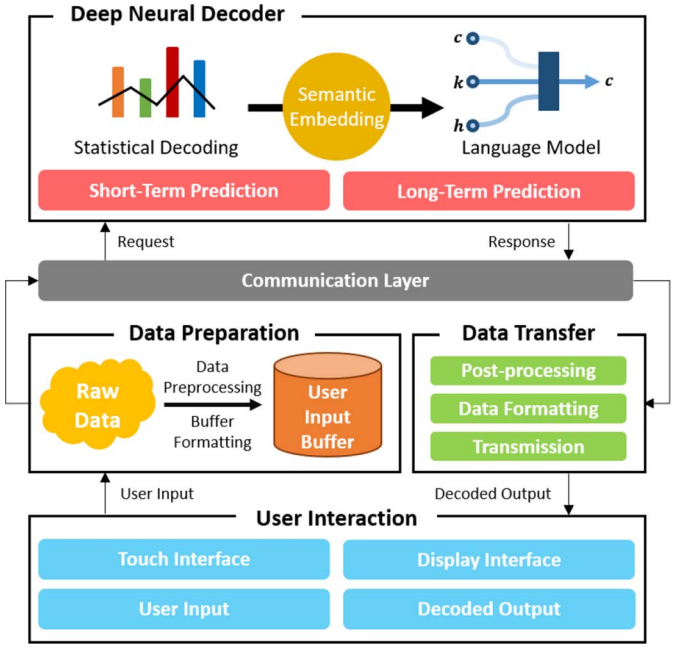

Fig. 6. I-Keyboard system architecture. The user interaction module accepts the user inputs and displays the decoding output. Data preparation and the data transfer modules function as a data pathway. DND translates the user inputs into character sequences.

frameworks hardly support a native implementation of DL algorithms for application frameworks with a few exceptions. The need for the communication layer will disappear when the support for the native implementation emerges. DND decodes the user input in both short-term and long-term manners. When a user starts to type, DND decodes in the short-term manner due to the vacancy of user input buffer. After the user has been typing for a little while, DND uses the long-term prediction scheme. The details of DND follow in the next section. The data transfer module receives the decoded sequence and lets the interaction module display the result.

B. Deep Neural Decoder

1) Problem Formulation: We formulate the keyboard decoding problem as follows. Given a set of touch points on the touch screen $\mathbb{T} = (t_1, \ldots, t_n)$, where $t_i = (x_i, y_i)$ consisting of x and y positions and n stands for the sequence length, a decoding algorithm finds the most probable sequence of characters $\hat{\mathbb{C}} = (\hat{c_1}, \ldots, \hat{c_n})$, where $\hat{c_i} \in D = \{e_1, \ldots, e_k\}$ for $1 \leq i \leq n$, D is a character dictionary, including space, enter, period, and apostrophe, and k is the size of the character dictionary

$$\hat{\mathbb{C}} = \arg\max_{\mathbb{C}} P(c_1, \ldots, c_n | t_1, \ldots, t_n) = \arg\max_{\mathbb{C}} P(\mathbb{C}|\mathbb{T}). \quad (3)$$

We use the maximum-likelihood estimation (MLE). By applying the Bayes rule, the right-hand side of (3) becomes

$$\arg\max_{\mathbb{C}} P(\mathbb{C}|\mathbb{T}) = \arg\max_{\mathbb{C}} \frac{P(\mathbb{T}|\mathbb{C}) \cdot P(\mathbb{C})}{P(\mathbb{T})}. \quad (4)$$

Removing the invariant $P(\mathbb{T})$ turns the decoding objective into

$$\hat{\mathbb{C}} = \arg\max_{\mathbb{C}} P(\mathbb{T}|\mathbb{C}) \cdot P(\mathbb{C}). \quad (5)$$

In this setting, $P(\mathbb{T}|\mathbb{C})$ and $P(\mathbb{C})$ refer to a user mental model and language model, respectively. Since considering the long-term dependency complicates the statistical model for $P(\mathbb{T}|\mathbb{C})$, the conventional models approximate the mental model by assuming the independence property and a Gaussian distribution. On the other hand, we model both the user mental model and the language model using bi-directional GRUs to handle the long-term dependency. By incorporating the bi-directional GRU units, the proposed decoder can translate the input touch points into a character sequence without compromising the performance with the model complexity.

2) Neural Architecture: Fig. 7 shows the neural architecture of DND. The user input flows from the input layer to the final output layer. The input layer concatenates the x and y components of the user input and transforms the user input into input vectors. In addition, the input layer receives the user input at multiple time steps for the statistical decoding layer and the character language model (CLM) statistically and semantically examines both the short-term and long-term dependencies. To support this, the input vectors are windowed and DND uses a certain number of past input vectors each time.

The statistical decoding layer receives the input vectors and translates them to a sequence of characters. The decoding layer does not take the semantic relations between characters into account, but the layer only considers positional relations among the user input during the translation. The decoding layer consists of a stack of bi-directional GRU units. Bi-directional GRUs can model the dependencies in forward and backward directions. Since I-Keyboard lets users type at any position on the touch interface, regarding both forward and backward dependencies is crucial for the decoding performance. We select GRUs instead of LSTMs for the sake of computational efficiency.

The softmax selection of the intermediate and the final output layers first maps the output from the GRU units to a tensor whose dimension matches the size of the character dictionary. The softmax selection utilizes a linear layer for the mapping. Then, the softmax selection applies the softmax function to the tensor and selects the most probable components as follows:

$$\hat{c_i} = \arg\max_{c} \frac{\exp(v_i^c)}{\sum_{j=1}^{k} \exp(v_i^j)} \quad (6)$$

where i $(1 \leq i \leq n)$ is the index of the character, v is the output vector of CLM, and k is the size of the character dictionary.

The acceleration layer enforces an additional auxiliary loss [26] for the intermediate output to boost the training procedure. The acceleration layer significantly improves the convergence speed as well as the decoding accuracy. Without the auxiliary loss, the DND architecture becomes too deep for the gradient caused by the final loss to travel through the network. In such case, the learning would not occur in the

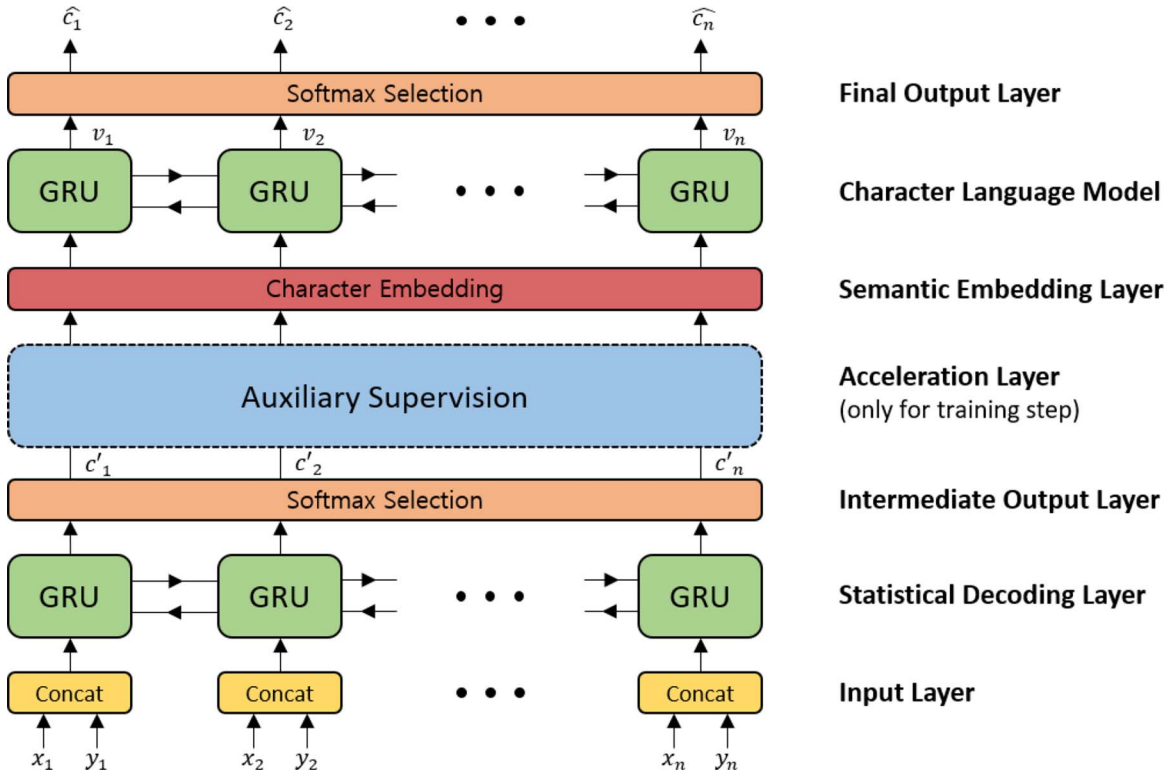

Fig. 7. DND architecture. The statistical decoding layer extracts the character sequences from positional information implicit to the user input. The CLM corrects the positional errors with semantic information. The acceleration layer facilitates the learning process.

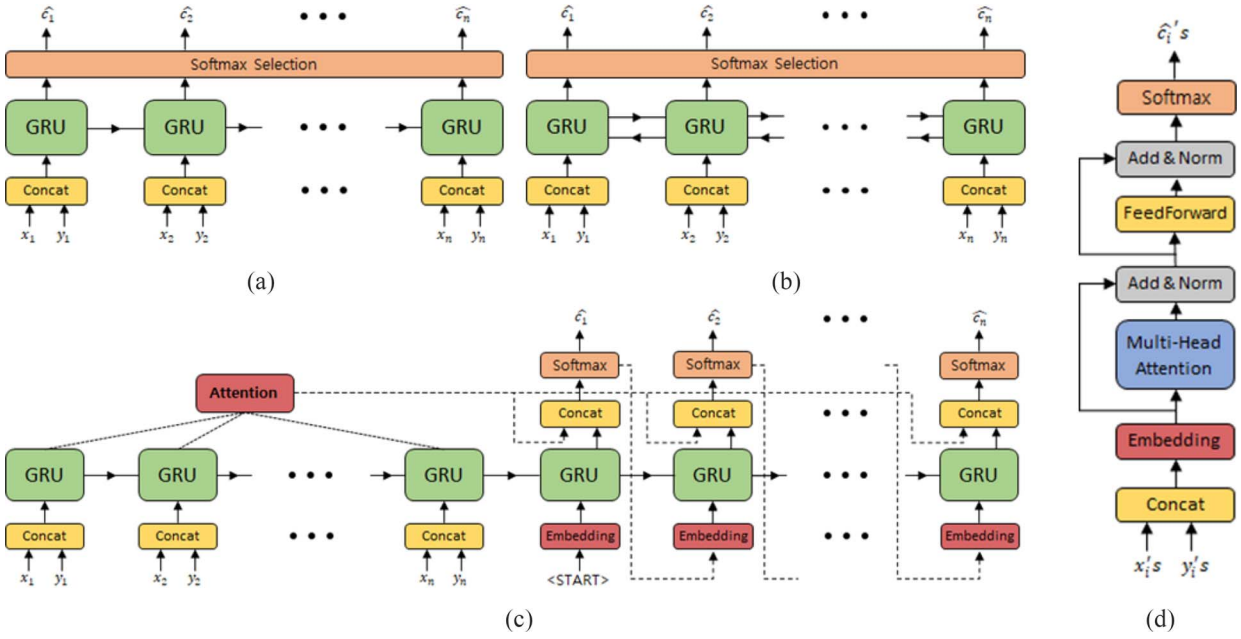

Fig. 8. Architectures of the baselines. For the transformer model, we only employ the encoder part to generate a sequence of characters from the user inputs. Other baselines do not vary from the basic architectures. (a) uni-rnn. (b) bi-rnn. (c) seq2seq. (d) transformer.

appropriate direction for the middle layers. We use the softmax cross-entropy loss for both the acceleration layer and the final output loss. The acceleration layer is enabled only for the training step.

Next, the semantic embedding layer embeds each character into a character vector [27]. The embedding vector stores the semantic relations between characters after training. The CLM revises the positional errors

entailed in the user input data using the semantic relations between characters. It learns how characters are related and how to correct typos. The CLM also employs bi-directional GRU units to handle both forward and backward dependencies.

V. Simulation

In this section, we focus on verifying the decoding accuracy of the proposed DND compared to a couple of baselines. Then, we present the ablation study results for various parameter choices.

A. Simulation Setting

1) Baselines: We verified the performance of DND in comparison with a few baseline methods. We did not include the conventional statistical models such as a Gaussian model [25] since they failed to work in the challenging I-Keyboard scenario. The positions of user hands vary over time and the position of each key is not restricted to a specific area in the scenario. Therefore, each key can present at any positions of touch devices and the Gaussian modeling becomes meaningless. We selected four models among the latest and most popular DL networks [28]: 1) RNN (uni-rnn); 2) bi-directional RNN (bi-rnn); 3) encoder–decoder attention (seq2seq) [29]; and 4) transformer (trans) [30] models. Fig. 8 shows the architectures of the baseline models. For the transformer model, we only used the encoder part to transform the user inputs into the sequence of characters.

2) Ablation Study: We explored various parameter choices for each model. We modified multiple parameters and investigated the effect of parameter modifications on the decoding performance. We controlled the following parameters: the numbers of stacks (2, 3, 4), units for GRU (32, 64), heads (8, 16 for trans models), and the use of auxiliary loss (for DND).

3) Data Preparation: We separated the collected data into three parts: 1) train; 2) validation; and 3) test datasets. We randomly chose two participants and assigned their data as the test dataset. For the validation dataset, we selected one participant. We randomly shuffled the train dataset for better convergence. In addition, we augmented the train dataset by adding random offsets in both horizontal and vertical directions. We repeated the data augmentation process five times. We used the validation dataset to prevent overfitting. We stopped the training procedure when the error rate on the validation dataset started to increase.

4) Evaluation Metrics: We assessed each decoding algorithm with three metrics: 1) decoding speed; 2) character error rate (CER); and 3) word error rate (WER). We measured the average decoding speed since ensuring a real-time operation is of great importance for decoders. We measured the accuracy with CER and WER. CER is defined as

$$\text{CER} = \frac{\text{MCD}(S, P)}{\text{length}_c(P)} \times 100\,(\%) \tag{7}$$

where $\text{MCD}(S, P)$ is the minimum character distance between the decoded phrase S and the ground-truth phrase P, and $\text{length}_c(P)$ is the number of characters in P. Similarly, WER is defined as

$$\text{WER} = \frac{\text{MWD}(S, P)}{\text{length}_w(P)} \times 100\,(\%) \tag{8}$$

where $\text{MWD}(S, P)$ is the minimum word distance between S and P, and $\text{length}_w(P)$ is the number of words in P. CER and WER count the number of insertions, deletions, and substitutions of characters or words to transform S into P.

5) Implementation Details: We implemented DND with Tensorflow (Python 3.6). We set the size of character embedding vector and the character dictionary as 16 and 31, respectively. We used the Levenshtein measure to calculate MCD and MWD. We set other model parameters as stated in Section V-A2. For training, we used Adam optimizer and set the initial learning rate as 0.001. We increased the learning rate when the minimum validation loss was updated and decreased the learning rate when the minimum validation loss was not updated. The rate of learning rate variation was 0.1. We stopped training when the minimum validation loss did

TABLE IV
Summary of Simulation Results

Model Name	Parameter	CER	WER	Time
uni-rnn	s2u32	11.21	41.27	0.72 msec
	s2u64	7.93	30.89	0.95 msec
	s3u32	9.19	35.32	0.94 msec
	s3u64	7.10	28.53	0.89 msec
	s4u32	7.15	29.22	1.12 msec
	s4u64	6.42	25.21	1.08 msec
bi-rnn	s2u32	6.22	22.85	0.88 msec
	s2u64	4.53	16.62	0.98 msec
	s3u32	4.08	15.24	1.11 msec
	s3u64	1.84	8.73	1.10 msec
	s4u32	4.08	16.62	1.28 msec
	s4u64	1.99	9.97	1.33 msec
seq2seq	s2u32	83.45	100	1.97 msec
	s2u64	54.43	68.98	2.04 msec
	s3u32	64.18	81.44	2.54 msec
	s3u64	15.82	18.14	2.58 msec
	s4u32	48.92	63.16	2.63 msec
	s4u64	8.59	9.97	2.69 msec
trans	s3u32h8	24.91	56.37	0.65 msec
	s3u32h16	28.16	64.68	0.71 msec
	s3u64h8	23.70	55.26	**0.63 msec**
	s3u64h16	23.98	55.40	0.74 msec
DND	s2u32	77.33	100	1.67 msec
	s2u32au	2.37	9.00	1.67 msec
	s2u64	68.19	100	1.72 msec
	s2u64au	**1.18**	**4.16**	1.65 msec
	s3u32	76.22	100	2.05 msec
	s3u32au	1.86	6.65	2.12 msec
	s3u64	75.26	100	2.06 msec
	s3u64au	1.54	5.12	2.10 msec
	s4u32	75.77	100	2.72 msec
	s4u32au	4.33	15.51	2.50 msec
	s4u64	77.36	100	2.65 msec
	s4u64au	1.26	4.43	2.72 msec

not change for 10 epochs. We chose models with the minimum validation loss for the simulation. We used one GPU (Titan X) for training and one CPU (2.9 GHz Intel Core i7) on MacBook Pro for testing.

B. Results and Analysis

Table IV summarizes the simulation results. The characters "s", "u", "h", and "au" in the parameter column indicate the numbers of stacks, units of GRU and heads, and the presence of auxiliary loss, respectively, and each number following the character specifies the dimension. CER, WER, and time (the average time needed to decode one word) in the table were evaluated on the test data. Table V displays the decoding examples of the bi-rnn and DND models compared to the ground truth.

Comparing the performance between models reveals the intuitions for designing DND. First, the results of uni-rnn and bi-rnn imply the importance of considering the backward dependencies in decoding performance. The superior performance of bi-rnn (s2u32, s3u32, s4u32) over uni-rnn (s2u64, s3u64, s4u64) proves the importance of the backward dependency since uni-rnn with 64 GRU units and bi-rnn with 32 GRU units have approximately the same model size. In addition, uni-rnn could not achieve high accuracy even when the model size increases.

Next, the superior performance of DND over bi-rnn suggests the limitation of decoding with only positional information. Though the bi-rnn models showed reasonable performance, they could not correct the positional mistakes made by users. By employing semantic embedding, DND further improved the performance of bi-rnn. In addition, the auxiliary loss forced by the acceleration layer lets gradients pass through the DND architecture. Without the auxiliary loss, DND becomes too deep for the single loss to train. The performance difference between DND with the auxiliary loss and DND without the loss verifies the effectiveness of the acceleration layer.

The seq2seq and trans models did not exhibit acceptable performance although they have demonstrated the state-of-the-art performance in other tasks such as neural translation. We presume the seq2seq and trans models require semantic information implicit in the input data to show high performance as reported. The input for the decoding task in this article rarely contained semantic information but only physical (positional) information. The lack of semantic meaning in the input data hindered the models to translate the input sequence to the character sequence.

All of the models guarantee a real-time operation with reasonable decoding speeds. We did not use a special computation device, but just a normal one. Moreover, we did not utilize a batch-based computation since DND accepts the user input one at a time, not in the batch-based form in the real-world setting. When DND accepts a batch of input, the throughput increases several times (up to the batch size). If DND is deployed in a server, the server can utilize the batch-based computation.

TABLE V
EXAMPLES OF THE DECODING RESULTS

Ex. 1	G.T.	do you know where there's a store that sells towels.
	bi-rnn	do you know where there's a store that sells towels.
	DND	do you know where there's a store that sells towels.
Ex. 2	G.T.	can i have a receipt please.
	bi-rnn	can i have a receilt please.
	DND	can i have a receipt please.
Ex. 3	G.T.	my throat is sore.
	bi-rnn	my thrat his sore.
	DND	my theat ois sore.

VI. EXPERIMENTAL VERIFICATION

To demonstrate the performance of I-Keyboard in real application settings, we designed an experiment incorporating users. We report the typing speed and the accuracy as performance metrics.

A. Experiment Setting

1) Participants: Thirteen skilled QWERTY keyboard users (5 males and 8 females) from the campus who had not joined in the data collection process participated in the experiments. Their ages range from 22 to 32 (average = 25.6, std = 2.5). All subjects responded that they used on daily basis both physical keyboards and soft keyboards on mobile devices and they could type in an eyes-free manner.

2) Apparatus: We conducted the experiments using the same apparatus of the data collection process. In addition, we measured the time spent on typing the task phrases. Other than that we used the same interface as in the data collection procedure.

3) Baselines: We did not select a baseline method for comparison in the experiments for the following reasons. First, we pioneered the I-Keyboard scenario in which users type using ten fingers in an eyes-free manner with no predefined region for typing and no calibration step. As this scenario is first attempted in this article, this type of method has not been mentioned in the literature. Second, there is no probability that the conventional methods with an enhanced calibration may perform better. The I-Keyboard setting reduces the constraints other methods impose, such as a predefined typing region and a calibration step. Without such constraints, the decoding becomes statistically more difficult since the touch points spread over a larger area and vary significantly point after point due to hand position shifts. Third, we designed I-Keyboard through comprehensive simulations which already compared various model architectures and parameters. We only referred to the performance metrics of the previous state-of-the-art [17] to assess the performance of I-Keyboard, since the accuracy of the state-of-the-art in the challenging I-Keyboard setting was only 32.37% (CER).

4) Performance Metrics: We evaluated the I-Keyboard performance with the following metrics: speed in WPM, accuracies in CER and WER, and subjective rates in 5-Likert

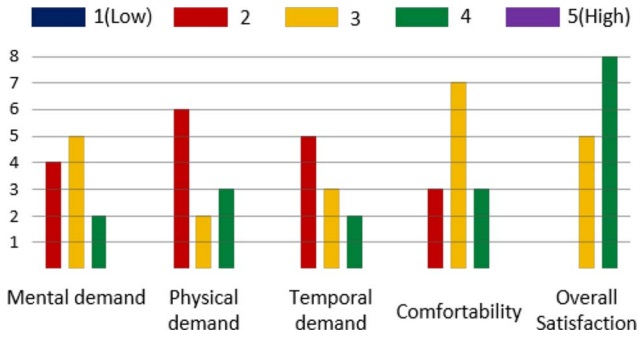

Fig. 9. Subjective ratings from the participants. The result attests that users can type with little mental and physical fatigue.

scale. WPM is defined as follows:

$$\text{WPM} = \frac{|\text{length}_c(S) - 1|}{M} \times \frac{1}{n_c} \tag{9}$$

where M is the time elapsed from the first keystroke to the last keystroke for S in minutes, and n_c is the average number of characters in a word. We set $n_c = 5$ in the analysis. We gathered subjective feedback from the participants in five perspectives: mental, physical and temporal demands, comfortability, and overall satisfaction.

5) Procedure: We first collected the demographic information from the participants. Then, we informed them of the experiment procedure. The experiments consisted of two phases: typing with a physical keyboard and I-Keyboard. To remove the effect of bias, we alternated the order of typing devices from participant to participant. We instructed the participants to type as accurately and quickly as possible. For each keyboard, we first presented ten phrases randomly sampled from the phrase set as a warm-up session. After participants became familiar with each keyboard, we required each participant to transcribe 20 phrases randomly selected from the phrase set. Finally, the participants took a survey to rate I-Keyboard subjectively.

B. Results and Analysis

Fig. 9 and Table VI summarize the experimental result. The average typing speeds of the physical keyboard and I-Keyboard were 45.57 and 51.35 WPM, respectively. The average speed of the physical keyboard measured in this article is slower than the previously reported average speed (55 WPM) [17]. We investigated the typing speed of each participant and some of them showed fairly slow typing speeds with both the physical keyboard and I-Keyboard. The minimum typing speeds for each typing interface are 35.13 and 26.88 WPM, respectively. This factor lowers the average typing speeds of both interfaces. However, the typing speed of I-Keyboard is faster than the previous state-of-the-art (41.4 WPM), though the average speed of I-Keyboard might have been reduced.

In addition, we report a much higher speed ratio between the typing interfaces (45.57/51.35 = 88.74%) than the baseline (41.4/55.5 = 74.59%). We infer from the user feedback that the enhanced typing speed resulted from the increased freedom provided by I-Keyboard. The users can type anywhere on the touch screen without worrying about key positions and typos. The participants replied that this feature makes I-Keyboard convenient. In fact, we can further improve the typing speed of I-Keyboard with more sensitive touch devices. The participants tried to type faster but the device in this article could not sense the touches when the typing speed exceeded the sensitivity of the device. We expect I-Keyboard would offer a more comfortable and faster typing interface with careful choice of the touch device.

The overall accuracies of I-Keyboard in CER and WER are 98.91% and 96.12%, respectively. The accuracy is slightly higher than the accuracy of the test set in simulation. We assume the participants attempted to type as precisely as possible following the instruction given before the experiment. We instructed the participants to type as precisely and swiftly as possible. For the data collection process, we did not inform such instruction.

The subjective rating from the participants reveals key features of I-Keyboard. The physical and temporal demands are left skewed, meaning the users can type quickly without experiencing physical fatigue. This implies improved typing speed and user experience. The mental demand and comfortability approximately show the Gaussian distributions signifying these metrics require further investigation to extract meaning. We suspect the participants do not take typing as a burden. The overall satisfaction is 3.62.

VII. Discussion

I-Keyboard implements an eyes-free ten-finger typing interface that does not involve a calibration procedure. Moreover, the users do not need to learn any new concept regarding I-Keyboard prior to usage. They can just start typing naturally by transferring the knowledge of physical keyboards. The users can keep typing even when they have taken off their hands phrase after phrase without an additional calibration step. I-Keyboard targets mobile devices since most mobile devices (Android iOS) support QWERTY keyboards and users in general use them. However, there still exist a few remaining works for further improvements although we established the effectiveness of I-Keyboard.

First, we can further experiment with various touch screen sizes. In this article, we utilized a relatively large touch screen. By using the large touch screen, we intended to consider radical situations such as moving hands dynamically phrase after phrase. Even in such situations, I-Keyboard can decode the keystrokes with high accuracy. Thus, the users can both naturally and comfortably type with I-Keyboard. Furthermore, increasing display size is the trend for mobile devices and the trend will accelerate with the development of foldable screens [31]. The current version of I-Keyboard can already support smartphones with a few adjustments such as supporting typing on a QWERTY keyboard with two thumbs. Nevertheless, we plan to extend I-Keyboard to other touch screens in the succeeding study. When targeting multiple screen sizes, transfer learning [32] would enable one general platform for all.

TABLE VI
COMMENTS FROM USERS

Positive	"I expect I-Keyboard will show more excellent performance in terms of speed and comfort with better touch device."
	"It was nice that I can comfortably and freely position my hands."
	"I could hit the keyboard with my usual typing habit instead of typing it on the pre-defined keyboard."
Negative	"It would be better if the touch device can perceive fast touches"
	"Users can get confused with the keys in similar locations."
	"I-Keyboard could not recognize key press when I touch it with my fingernail."

We implemented I-Keyboard with the LG touch monitor using WPF. In the following research, we can extend the presented I-Keyboard environment to any flat surfaces with advanced touch sensing techniques. A depth-based touch detection system [33] could enable such extension. When I-Keyboard is integrated with advanced touch detection methods, a number of mobile systems would benefit from it. For example, I-Keyboard can offer an effective text-entry method for VR systems. Since the VR systems require an eyes-free text-entry method, I-Keyboard can be an attractive alternative for current text-entry tools for VR systems.

The next-generation I-Keyboard can support nonalphabetic characters (e.g., numbers, punctuation, and functional keys) for better user experience, though the users can type lowercase alphabets, apostrophe, enter, space, and period with the current I-Keyboard. Specifically, the shift key can be detected by a multitouch detection algorithm by recognizing the first key pressed while the second key is stroked as the shift key. A gesture detection algorithm would enable the delete key by assigning a swipe gesture as the delete key. In addition, the users would not need to wear gloves with a multitouch detection algorithm by detecting palms.

Moreover, the next-generation I-Keyboard would instantly display output characters even when users start to type with I-Keyboard. For this, we would come up with enhanced user-buffer processing or an improved deep neural network model. Currently, the users have to type at least the size of the user buffer to view the typed text at the initial phase, though they can view the typed text instantly after the buffer is filled. This could puzzle users when they first try to use I-Keyboard. To further improve user experience, we can design an effective deep neural network model or efficient user-buffer architecture. We plan to develop the above-mentioned features for the next-generation I-Keyboard.

VIII. CONCLUSION

In this article, we proposed I-Keyboard with DND. I-Keyboard, for the first time, attempted to realize a truly imaginary keyboard that does not require calibration. The users can start typing from anywhere on the touch screen without being concerned with the keyboard shape and location in an eyes-free manner. In addition, users do not need to learn anything before typing. For the development of I-Keyboard, we conducted a user study while collecting the largest dataset. We analyzed the user behaviors in the eyes-free ten-finger scenario which enforces minimum constraints. The user mental models consistently displayed similar layouts as a physical keyboard, though the models drift in location and vary in shape over time. After confirming the feasibility, we designed I-Keyboard and DND. We utilized a deep neural architecture to handle the dynamic variation of the user mental models and the semantic embedding technique to further boost the decoding performance. Simulations and experiments verified the superior performance of I-Keyboard. The accuracy of DND was 95.84% and 96.12% in the simulation and the experiment, respectively, surpassing the performance of the baseline by 4.06%. The users can type at 45.57 WPM (88.74% of the typing speed with a physical keyboard) breaking the previous state-of-the-art performance (41.4 WPM, 74.6%). We believe that I-Keyboard would create new value for the mobile computing industry since it could transform the way users enter text and related applications.

REFERENCES

[1] P. O. Kristensson, "Next-generation text entry," *IEEE Comput.*, vol. 48, no. 7, pp. 84–87, Jul. 2015.
[2] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 450–465, Feb. 2018.
[3] R. Kumar and P. Chaudhary, "User defined custom virtual keyboard," in *Proc. IEEE Int. Conf. Inf. Sci. (ICIS)*, 2016, pp. 18–22.
[4] F. C. Y. Li, L. Findlater, and K. N. Truong, "Effects of hand drift while typing on touchscreens," in *Proc. Graph. Interface*, 2013, pp. 95–98.
[5] T. V. Raman and C. L. Chen, "Eyes-free user interfaces," *IEEE Comput.*, vol. 41, no. 10, pp. 100–101, Oct. 2008.
[6] F. C. Y. Li, R. T. Guy, K. Yatani, and K. N. Truong, "The 1line keyboard: A qwerty layout in a single line," in *Proc. 24th Annu. ACM Symp. User Interface Softw. Technol.*, 2011, pp. 461–470.
[7] V. Fuccella, M. De Rosa, and G. Costagliola, "Novice and expert performance of keyscretch: A gesture-based text entry method for touchscreens," *IEEE Trans. Human–Mach. Syst.*, vol. 44, no. 4, pp. 511–523, Aug. 2014.
[8] T. Abuhmed, K. Lee, and D. Nyang, "UOIT keyboard: A constructive keyboard for small touchscreen devices," *IEEE Trans. Human–Mach. Syst.*, vol. 45, no. 6, pp. 782–789, Dec. 2015.
[9] S. Poularakis and I. Katsavounidis, "Low-complexity hand gesture recognition system for continuous streams of digits and letters," *IEEE Trans. Cybern.*, vol. 46, no. 9, pp. 2094–2108, Sep. 2016.
[10] M. Devanne, H. Wannous, S. Berretti, P. Pala, M. Daoudi, and A. D. Bimbo, "3-D human action recognition by shape analysis of motion trajectories on Riemannian manifold," *IEEE Trans. Cybern.*, vol. 45, no. 7, pp. 1340–1352, Jul. 2015.
[11] N. Yang and A. D. Mali, "Modifying keyboard layout to reduce finger-travel distance," in *Proc. IEEE 28th Int. Conf. Tools Artif. Intell. (ICTAI)*, 2016, pp. 165–168.
[12] S. Sarcar, J. P. P. Jokinen, A. Oulasvirta, Z. Wang, C. Silpasuwanchai, and X. Ren, "Ability-based optimization of touchscreen interactions," *IEEE Pervasive Comput.*, vol. 17, no. 1, pp. 15–26, Jan.–Mar. 2018.

[13] K. Ushida, Y. Sekine, and S. Hasegawa, "IPPITSU: A one-stroke text entry method for touch panels using Braille system," in *Proc. IEEE 3rd Glob. Conf. Consum. Electron. (GCCE)*, Oct. 2014, pp. 374–375.
[14] A. Gupta, M. Anwar, and R. Balakrishnan, "Porous interfaces for small screen multitasking using finger identification," in *Proc. 29th Annu. Symp. User Interface Softw. Technol.*, 2016, pp. 145–156.
[15] K. P. Fishkin, T. P. Moran, and B. L. Harrison, "Embodied user interfaces: Towards invisible user interfaces," in *Proc. Eng. Human–Comput. Interact.*, 1998, pp. 1–18.
[16] K. Lyons, D. Plaisted, and T. Starner, "Expert chording text entry on the twiddler one-handed keyboard," in *Proc. IEEE 8th Int. Symp. Wearable Comput.*, vol. 1, 2004, pp. 94–101.
[17] W. Shi, C. Yu, X. Yi, Z. Li, and Y. Shi, "TOAST: Ten-finger eyes-free typing on touchable surfaces," in *Proc. ACM Interact. Mobile Wearable Ubiquitous Technol.*, vol. 2, no. 1, 2018, p. 33.
[18] J. Clawson, K. Lyons, T. Starner, and E. Clarkson, "The impacts of limited visual feedback on mobile text entry for the twiddler and mini-QWERTY keyboards," in *Proc. 9th IEEE Int. Symp. Wearable Comput.*, 2005, pp. 170–177.
[19] R. Rosenberg and M. Slater, "The chording glove: A glove-based text input device," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 29, no. 2, pp. 186–191, May 1999.
[20] I. Almusaly, R. Metoyer, and C. Jensen, "Evaluation of a visual programming keyboard on touchscreen devices," in *Proc. IEEE Symp. Vis. Lang. Human Centric Comput. (VL/HCC)*, 2018, pp. 57–64.
[21] I. S. MacKenzie, S. X. Zhang, and R. W. Soukoreff, "Text entry using soft keyboards," *Behav. Inf. Technol.*, vol. 18, no. 4, pp. 235–244, 1999.
[22] L. Findlater, J. O. Wobbrock, and D. Wigdor, "Typing on flat glass: Examining ten-finger expert typing patterns on touch surfaces," in *Proc. ACM SIGCHI Conf. Human Factors Comput. Syst. (CHI)*, 2011, pp. 2453–2462.
[23] Y. Lu, C. Yu, X. Yi, Y. Shi, and S. Zhao, "BlindType: Eyes-free text entry on handheld touchpad by leveraging thumb's muscle memory," in *Proc. ACM Interact. Mobile Wearable Ubiquitous Technol.*, vol. 1, no. 2, 2017, pp. 1–24.
[24] X. Yi, C. Yu, W. Shi, and Y. Shi, "Is it too small? Investigating the performances and preferences of users when typing on tiny qwerty keyboards," *Int. J. Human–Comput. Stud.*, vol. 106, pp. 44–62, Oct. 2017.
[25] S. Zhu, T. Luo, X. Bi, and S. Zhai, "Typing on an invisible keyboard," in *Proc. CHI Conf. Human Factors Comput. Syst.*, 2018, p. 439.
[26] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 2881–2890.
[27] H. Zhang, J. Li, Y. Ji, and H. Yue, "Understanding subtitles by character-level sequence-to-sequence learning," *IEEE Trans. Ind. Informat.*, vol. 13, no. 2, pp. 616–624, Apr. 2017.
[28] T. Young, D. Hazarika, S. Poria, and E. Cambria, "Recent trends in deep learning based natural language processing [review article]," *IEEE Comput. Intell. Mag.*, vol. 13, no. 3, pp. 55–75, Aug. 2018.
[29] T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," in *Proc. Conf. Empirical Methods Nat. Lang. Process.*, 2015, pp. 1412–1421.
[30] A. Vaswani *et al.*, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5998–6008.
[31] W.-J. Wu *et al.*, "High-resolution flexible AMOLED display integrating gate driver by metal–oxide TFTs," *IEEE Electron Device Lett.*, vol. 39, no. 11, pp. 1660–1663, Nov. 2018.
[32] D. Wang *et al.*, "Softly associative transfer learning for cross-domain classification," *IEEE Trans. Cybern.*, to be published.
[33] A. D. Wilson, "Using a depth camera as a touch sensor," in *Proc. ACM Int. Conf. Interact. Tabletops Surfaces*, 2010, pp. 69–72.

Ue-Hwan Kim received the B.S. and M.S. degrees in electrical engineering from the Korea Advanced Institute of Science and Technology, Daejeon, South Korea, in 2013 and 2015, respectively, where he is currently pursuing the Ph.D. degree.

His current research interests include service robot, human robot interaction, cognitive Internet of Things, human computer interaction, computational memory systems, and learning algorithms.

Sahng-Min Yoo received the B.S. degree in electrical engineering from the Korea Advanced Institute of Science and Technology, Daejeon, South Korea, in 2017, where he is currently pursuing the integrated master's and Doctoral degree.

His current research interests include user experience, service robot, and DL algorithms.

Jong-Hwan Kim (F'09) received the Ph.D. degree in electronics engineering from Seoul National University, Seoul, South Korea, in 1987.

Since 1988, he has been with the School of Electrical Engineering, Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea, where he is leading the Robot Intelligence Technology Laboratory as a KT Endowed Chair Professor, and the Director for KoYoung-KAIST AI Joint Research Center, and Machine Intelligence and Robotics Multi-Sponsored Research and Education Platform. He has authored 5 books and 5 edited books, 2 journal special issues and around 400 refereed papers in technical journals and conference proceedings. His research interests include intelligence technology, machine intelligence learning, and AI robots.