Type Anywhere You Want: An Introduction to Invisible Mobile Keyboard

Sahng-Min Yoo*, Ue-Hwan Kim*, Yewon Hwang and Jong-Hwan Kim
KAIST, Republic of Korea
{smyoo, uhkim, ywhwang, johkim}@rit.kaist.ac.kr

Abstract
Contemporary soft keyboards possess limitations: the lack of physical feedback results in an increase of typos, and the interface of soft keyboards degrades the utility of the screen. To overcome these limitations, we propose an Invisible Mobile Keyboard (IMK), which lets users freely type on the desired area without any constraints. To facilitate a data-driven IMK decoding task, we have collected the most extensive text-entry dataset (approximately 2M pairs of typing positions and the corresponding characters). Additionally, we propose our baseline decoder along with a semantic typo correction mechanism based on self-attention, which decodes such unconstrained inputs with high accuracy (96.0%). Moreover, the user study reveals that the users could type faster and feel convenience and satisfaction to IMK with our decoder. Lastly, we make the source code and the dataset public to contribute to the research community.

1 Introduction
Text-entry plays a crucial role in Human-Computer Interaction (HCI) applications. It offers an effective and efficient way for humans to deliver messages to computers. In the early stage of text-entry research, physical keyboard-based text-entry methods were dominant. As mobile technology integrates into daily interactions in people’s lives, the need for a text entry system that suits contemporary mobile devices has emerged. In the process, soft keyboards have become the gold standard of text-entry methods for mobile devices. Soft keyboards do not require additional hardware, which improves mobility and provides an intuitive interface that guarantees usability.

However, current soft keyboards reveal some limitations: high rate of typos, lack of tactile feedback, inconsideration of users’ different keyboard mental model, and large screen occupation. Since there are no clear boundaries between keys, typos commonly occur in soft keyboards by pressing an adjacent key by accident. Lack of tactile feedback requires the user’s constant monitoring which could result in physical strain. Furthermore, all users are forced to type according to the given standard keyboard without any customization, although each person has a different keyboard mental model depending on their physical structure or typing habits. Lastly, with the development of Information and Communication Technology (ICT), the amount of information that smartphones can provide to humans has exploded, but soft keyboards ironically cover about half of the screens, obstructing the graphical user interface rather than delivering more information compactly.

To overcome the limitations mentioned above, we propose Invisible Mobile Keyboard (IMK) with our baseline solution, Self-Attention Neural Character Decoder (SA-NCD), and collect the most extensive dataset of text-entry studies to train the proposed decoder. First of all, the proposed IMK is invisible; the entire screen can be used for displaying content which enhances the interaction between human and mobile devices; it lets users freely type text anywhere on the screen according to their typing habits. By fulfilling the IMK decoding task, users can start typing on any position at any angle as they wish.

*Equal Contribution

Figure 1: Various user mental models when typing on Invisible Mobile Keyboard. Each sub-figure is drawn with collected data of each participant.
The proposed decoder for the IMK task, SA-NCD, actual-
izes the full customization of soft keyboards. SA-NCD first
decodes user inputs through a geometric decoder. The de-
coded result at this step could contain typos since users’ men-
tal models would not precisely match the conventional key-
board layout. Then, SA-NCD semantically analyzes the user
inputs (the decoded result of the geometric decoder) and cor-
rects the typos using a language model.

In summary, the main contributions of our work are as fol-
lows:

- We have collected a large-scale, richly annotated IMK
dataset which contains approximately 2M pairs of touch
points and text.
- We analyze the characteristics of user behavior on the
new proposed task in-depth.
- We propose a novel deep neural architecture, Self-
Attention Neural Character Decoder (SA-NCD) as a
baseline decoder for IMK systems.
- We conduct a user study with the IMK prototype using
our SA-NCD as a proof-of-concept to identify users’ de-
mand for research in this area.
- To contribute to the corresponding research community,
we make the source code of SA-NCD and the collected
data public. The source code and data are available at
https://github.com/sahngmin/InvisibleMobileKeyboard.

2 Related Work

Text entry systems of mobile devices thus far use various de-
coding methods, where the methods can be largely divided
into two categories: statistics-based and deep-learning-based
methods. For statistical decoding methods, generally a proba-
bilistic model is adopted. For instance, in [Findlauer and Wob-
brock, 2012], a Naive Bayes classifier was implemented to
decode user input from an adaptive keyboard where the layout
of the keyboard is altered to personalize to users. In [Vertanen
et al., 2015], a sentence-based decoding method was used to
automatically correct typos after the entire sentence was en-
tered. They adopted a probabilistic model for their decoder,
computing the likelihood of each character using the Gauss-
ian distribution of each key. In [Lu et al., 2017], the absolute
algorithm and the relative algorithm were proposed to enable
eyes-free typing, where the absolute algorithm decodes user
input based on the absolute position of touch endpoints, while
the relative algorithm decodes based on the vectors between
successive touch endpoints. Later, [Shi et al., 2018] enabled
eyes-free typing on a touch screen by applying a Markov-
Bayesian algorithm for input prediction, which considers the
relative location between successive touch points within each
hand respectively. To allow typing on an invisible keyboard
on mobile devices, [Zhu et al., 2018] proposed a decoding
mechanism which consists of a spatial model and a decoder.
The spatial model links a touch point to a key, while the de-
coder predicts the next word given the history of previous
words.

Recently, with the rise of deep learning, [Ghosh and Krist-
tessson, 2017] developed a sequence-to-sequence neural at-
tention network model for automatic text correction and com-
pletion. This was achieved by a combination of a character-
level CNN and a gated recurrent unit (GRU) [Chung et al.,
2014] encoder along with and a word-level GRU attention
decoder. In addition, I-Keyboard [Kim et al., 2019], another
deep learning-based text entry method, was proposed. For
I-Keyboard, Deep Neural Decoder (DND) was proposed as
a decoding algorithm which consists of two bidirectional
GRUs: one for a statistical decoding layer and the other for a
language model. The decoding algorithm we propose in this
work is deep-learning-based and employs not only a statisti-
cal model with masking process, but also a semantic model
to understand the relationship between characters.

3 Dataset

3.1 Dataset Construction

To collect data, we first implemented a simple web user inter-
face which can be accessed from any mobile device. Participants
used their own mobile devices to type given phrases on
the interface. The details of the data collection procedure are
described in the supplementary material. Our dataset contains
user initial, age, type of mobile devices, size of the screen,
time taken for typing each phrase, and annotation of typed
phrases with coordinate values of the typed position (x and y
points). The dataset we collected is the first and only dataset
for a novel IMK decoding task. Further, we have collected
significantly more data compared to the datasets used in the
existing text-entry research (see Table 1).

3.2 Dataset Analysis

We analyzed the user behavior when typing on IMK and iden-
tify the characteristics of the collected data.

Variance

We calculated the standard score (z-normalization) of x and
y coordinates by averaging the entire participants and charac-
ters over typing index t. The standard score of x and y posi-
tions, $z_x(t)$ and $z_y(t)$, are calculated as follows:

$$z_x(t) = \frac{1}{\sum_{i=1}^{N} \sum_{j=1}^{n_i} f(t, i, j) \sum_{i=1}^{N} \sum_{j=1}^{n_i} \frac{x_{i,j} - \bar{x}_{i,j}}{\sigma_{i,j}}},$$

(1)

with

$$f(t, i, j) = \begin{cases} 1, & \text{if } t \leq m_{i,j} \\ 0, & \text{otherwise} \end{cases}$$

Table 1: Comparison of dataset sizes for text-entry research. ‘#’ and ‘avg.’ stand for the number of participants and an average value, respectively.
Figure 2: Average 'q' (left) and 'p' (right) positions of individual participants. Dots of the same color in left and right sub-figure are from the same person.

Figure 3: $z_x(t)$ and $z_y(t)$ value over typing time index $t$, when typing with the IMK. Values bigger than four were cut off and not drawn.

Table 2: Statistics of collected data in x-axis and y-axis directions.

|            | mean  | std. dev. | min   | max   |
|------------|-------|-----------|-------|-------|
| scale x    | 0.99  | 0.27      | 0.36  | 3.12  |
| scale y    | 0.95  | 0.28      | -1.15 | 1.78  |
| offset x   | -2.00 | 25.68     | -180.88 | 221.90 |
| offset y   | -7.13 | 29.44     | -161.85 | 161.05 |

where $v$ is the vocabulary size, $N$ is the number of participants, $x_{i,j}^t$ is the x coordinate when $i$-th participant typing $j$-th character for $t$-th time, $\overline{x_{i,j}}$ is the mean value of $x_{i,j}^t$ taken for all $t$, and $n_{i,j}$ is the maximum number of times the $j$-th character appears for the $i$-th participant. $z_y(t)$ is calculated in the same way as $z_x(t)$, but with y coordinates. Both $z_x(t)$ and $z_y(t)$ fluctuate and the fluctuations get larger over time in both vertical and horizontal directions (see Figure 3).

Physical Dimensions
We examined the physical dimensions of user mental models to analyze how widely and actively users moved their fingers to type on IMK. First, we analyzed the positions of 'q' and 'p' since they are the farthest keys from the space bar. The average positions of 'q' and 'p' for each user were inconsistent and varied significantly (see Figure 2). Some users typed 'q' and 'p' at an average of only 50 pixels apart on the x-axis, while others typed at a distance of 300 pixels.

In addition, we calculated statistics of the physical dimensions through the average distance between 'space' and 'p' in each sentence. Since 'p' appeared 23.9 times more than 'q' in our dataset, 'p' was adopted for meaningful statistical analysis. We omitted the sentences that did not include 'p' for the analysis. Moreover, we use the average positions of 'space' and 'p' for the sentences where 'space' and 'p' appeared more than once. Table 2 summarizes the mean, standard deviation (std. dev.), min and max values of the scale, and offset of user mental models.

The first distance between ‘space’ and ‘p’ was defined as scale 1, and the offset was calculated by taking the difference between the average position of the first 10 inputs and the last 10 inputs. From the statistics, it can be seen that the users type on IMK with their own mental keyboards of inconsistent scales, and the offset over time is large. The fact that the minimum of the y-axis scale is negative means that data also includes some typos such that ‘p’ key is entered below the space bar.

As the analysis suggests, decoding our IMK dataset is a challenging task. Additionally, as Figure 1 shows, characters are not clustered neatly and typing patterns differ from user to user. Plus, the typing patterns of the right and left hands were distinguishable. Lastly, the distribution of the data consistently changed over time (see Figure 3) and the scale of the keyboard size was undefinable (see Figure 2).

4 A Baseline Approach for IMK Decoder

4.1 Problem Formulation

We propose a novel baseline approach for the IMK decoding task. Specifically, taking a sequence of coordinate values as input, we aim to decode a phrase which users intended to type on an invisible layout. Let $\hat{C} = \{\hat{c}_1, ..., \hat{c}_n\}$ denote a predicted character sequence by the decoding algorithm. The decoder aims to find the character sequence, $\hat{C}$, with the highest conditional probability given a typing input,
Figure 4: Network architecture of Self-Attention Neural Character Decoder. Q, K, and V stand for Query, Key, and Value of self-attention from Transformer.

$$T = \{t_1, t_2, ..., t_n\},$$ as follows:

$$\hat{C} = \arg \max_{\hat{c}_t} P(\hat{c}_1, ..., \hat{c}_n | t_1, ..., t_n). \quad (2)$$

4.2 Network Architecture

Our baseline approach, Self-Attention Neural Character Decoder (SA-NCD), consists of two decoder modules: 1) a geometric decoder ($G$) with its parameters $\phi_G$ and 2) a semantic decoder ($S$) with its parameters $\phi_S$. For an input sequence $X = \{x_1, x_2, ..., x_n\}$, let $G(X; \phi_G)$ and $S(X; \phi_S)$ denote the output from the geometric decoder and the semantic decoder, respectively. The geometric decoder takes in a touch input sequence and converts it into a character sequence by using the touch locations in the invisible keyboard layout. Then, the semantic decoder corrects decoding errors in the character sequence estimated by the geometric decoder while considering semantic meanings. Here, the confidence masking process determines the input of the semantic decoder by finding locations of the errors produced by the geometric decoder. The errors could be simply a case of incorrect predictions by the geometric decoder, or a case of absolute typos made by a user. One thing to note is that SA-NCD decodes the entire sequence input up to that point (not merely the most recent touch input). Therefore, the characters decoded in the previous steps are not fixed but can be changed by reflecting the semantic dependence.

**Geometric Decoder**

We use Bidirectional GRU (BiGRU) for the geometric decoder. To overcome the vanishing gradient problem, we adopt GRU as a recurrent neural network. In our task, even if a user enters the same character index, the input (the touch points) is different every time. Since the inputs are not a definite character index, the recurrent network suffers from the instability accumulation if it only proceeds in uni-direction. Thus, we take the advantage of forward and backward passes of BiGRU to overcome this issue.

**Confidence Masking**

The length of $G(X; \phi_G) = \{o_1, o_2, ..., o_n\}$ is the same as the input length, and each element has the dimension size of the vocabulary size ($v$). When considering the $i$-th input ($1 \leq i \leq n$) from a sequence, the probability (confidence) of the $k$-th character in the vocabulary, $p_{ki}^i$, can be calculated by the softmax function as follows:

$$\text{Softmax}(G(X; \phi_G)) = \{ (p_1^i, p_2^i, ..., p_v^i) \}_{i=1}^n, \quad (3)$$

with

$$p_{ki}^i = \frac{\exp(o_{ji}^i)}{\sum_{j=1}^v \exp(o_{ji}^i)},$$

where $o_{ji}^i$ is the value of $j$-th element of $o_i$ in $G(X; \phi_G)$ and $v$ is the vocabulary size.

At this step, the touch input detected at an ambiguous location that is difficult to decode with only geometric information results in low confidence. We replace the positions with a confidence lower than $\tau$ with mask tokens to provide the masked character sequence as input to the semantic decoder (confidence masking process). The following $CM$ and $Mask$ refer to the confidence masking process and the individual masking function as follows:

$$CM(\{x_1, ..., x_n\}) = \{\text{Mask}(x_1), ..., \text{Mask}(x_n)\}, \quad (4)$$

with

$$\text{Mask}(x_i) = \begin{cases} \text{Embed}(\arg \max_j p_{ji}^i), & \text{if } \max_j p_{ji}^i \geq \tau \\ \text{Embed}(j_{\text{mask}}), & \text{otherwise} \end{cases},$$

$$\text{Embed}(x) = \begin{cases} x \cdot \text{dim}, & \text{if } x \neq \text{mask} \\ \text{dim}, & \text{otherwise} \end{cases}, \quad (5)$$

where $\text{dim}$ is the dimension size of the vocabulary.
where \( x_i = (p^1_i, p^2_i, \ldots, p^n_i) \) and \( f_{\text{mask}} \) is the index of mask token. \( \text{Embed} \) is an embedding layer which encodes index to an embedded vector.

**Semantic Decoder**

We use the Transformer [Vaswani et al., 2017] encoder architecture as a semantic decoder. The embedded hidden state that has passed the geometric decoder followed by confidence masking is processed by the semantic decoder with the self-attention mechanism of Transformer. Here, the semantic decoder acts as a character language model, replacing the masked characters with appropriate characters. Through the proposed network architecture (see Figure 4), the decoded character sequence \( \hat{C} \) in Equation (2) can be expressed as follows:

\[
\hat{C} = S(CM(\text{Softmax}(G(X; \phi_G))); \phi_S).
\] (5)

**Training Scheme**

We pre-train both the geometric decoder and the semantic decoder to enhance their respective roles. In particular, we pre-process One Billion Word Benchmark (1BW) [Chelba et al., 2014] to train the semantic decoder as a masked character language model by following the same pre-training scheme of BERT [Devlin et al., 2019]. Then, we fine-tune the two decoder modules together by first freezing \( \phi_G \) and only training \( S(X; \phi_S) \), and then training \( G(X; \phi_G) \) while freezing \( \phi_S \) repeatedly.

5 Results

5.1 Experiment Settings

**Data Split**

Of the 88 participants of our study, 78 participants collected data using 300 sentences of 1BW, and each five of remaining participants collected data using 100 phrases of MacKenzie set (MacK) [MacKenzie and Soukoreff, 2003] and 1,000 most commonly used English phrases set (Common) [EnglishSpeak, 2019], respectively. Of these, we only utilized the data collected from 1BW data to train our decoding algorithm. We used the remaining MacK and Common sets for evaluation purposes to assess whether the proposed decoder performs adequately on sentences it has never seen before.

We divided the 1BW data into a train (from 70 participants), validation (from 3 participants), and test (from 5 participants) sets. We divided the dataset by person rather than combining all the datasets and dividing it by ratio. This split method would verify if SA-NCD can decode sentences even from a new person with an unseen mental model.

**Metrics**

We used CER (Character Error Rate) and WER (Word Error Rate) as evaluation metrics defined as follows:

\[
\text{CER} = \frac{\text{MCD}(\hat{C}, P)}{\text{length}_a(P)} \times 100\%.
\] (6)

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

\[
\text{WER} = \frac{\text{MWD}(\hat{C}, P)}{\text{length}_w(P)} \times 100\%,
\] (7)

where \( \text{MWD}(\hat{C}, P) \) is the minimum word distance between \( \hat{C} \) 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 \( \hat{C} \) into \( P \).

**Compared Models**

We verified the performance of the proposed baseline SA-NCD compared to several sequential neural models. We implemented representative deep neural networks designed for sequential data: GRU, LSTM, Transformer, and I-keyboard which is the state-of-the-art method of the invisible keyboard task.

We measured the performance of various networks by varying the sizes which is denoted as \((L, d_h)\), where \( L \) is the number of stacked layers and \( d_h \) is the size of the hidden state. In order to thoroughly check the performance according to shallow to deep, narrow to wide networks, we adopted 4 and 12 for \( L \), and 256 and 512 for \( d_h \).

**Implementation Details**

We optimized \( \phi_G \) with SGD update, with the learning rate of 3.0 and gradient clipping at 0.5. For \( \phi_S \), we used the Adam optimizer with learning rate 7e-4. Threshold \( \tau \) for confidence masking was 0.45 and the vocabulary size \( v \) was 31. We minimized Cross-entropy loss to optimize the parameters. In addition, to quickly propagate information within the network, we employed the auxiliary losses for the character language model [Al-Rfou et al., 2019]. Moreover, we altered the position of layer normalization prior to self-attention to obtain the stability of the gradient and induce rapid convergence [Xiong et al., 2020].

We implemented deep neural network models with the open-source library PyTorch 1.4.0 version and used CUDA version 10.0 for the one GPU (GTX 1080ti) computation on a Ubuntu 16.04 workstation. We augmented our data by randomly shifting the x and y positions of each touch point by 3 or fewer pixels with a 50% probability. The validation accuracy was measured by using the validation set at every epoch, and if the maximum accuracy was not updated for more than 3 epochs, the training was terminated by early stopping.

5.2 Results and Analysis

**Decoding performance**

Table 3 shows the decoding performance (CER and WER) of our proposed baseline and the compared models with various network sizes. GRU showed better performance than LSTM in our task. Dealing with the gates of GRU is more advantageous rather than filtering the previous information with the cell state in the LSTM for the sequence data with geometric information. Since I-keyboard stacked two modules of bidirectional GRU, it performed better than a single GRU. Yet, I-keyboard also suffered from overfitting when stacked deeply, similar to GRU and LSTM. On the other hand, Transformer and SA-NCD could take advantage of greater complexity and
lead to higher performance without any overfitting when the model becomes deeper. This was because they utilized efficient multi-head self-attention. However, we found that the Transformer was incapable of extracting meaningful information in the IMK task. On the other hand, by separating the geometric decoder and the semantic decoder, SA-NCD showed the best performance. In particular, the proposed SA-NCD showed 5.51 and 7.71 CER on the MacK set and Common set, which consisted of unseen phrases from unknown users.

Table 3: CER and WER (values in parantheses) of decoding algorithms with various network size (Net. Size). Three sets used for performance evaluation are described in Section 5.1

| Decoding Algorithm | Net. Size | MacK | Common | Test |
|--------------------|-----------|------|--------|------|
| GRU                | 4 256     | 16.04| 21.51  | 15.76|
|                    |           | (11.52)| (15.69)| (9.77)|
|                    | 4 512     | 16.17| 21.96  | 16.90|
|                    |           | (11.84)| (16.18)| (10.80)|
|                    | 12 256    | 82.34| 82.19  | 82.28|
|                    |           | (73.93)| (68.70)| (85.86)|
|                    | 12 512    | 82.65| 82.71  | 82.35|
|                    |           | (71.85)| (67.70)| (85.17)|
| LSTM               | 4 256     | 46.10| 55.07  | 47.72|
|                    |           | (23.33)| (27.61)| (22.50)|
|                    | 4 512     | 53.48| 58.27  | 45.90|
|                    |           | (21.82)| (25.97)| (19.41)|
|                    | 12 256    | 80.61| 79.15  | 81.78|
|                    |           | (91.90)| (90.80)| (95.62)|
|                    | 12 512    | 80.73| 78.75  | 81.75|
|                    |           | (91.95)| (90.55)| (95.54)|
| I-keyboard         | 4 256     | 15.28| 18.14  | 12.52|
|                    |           | (9.91)| (12.00)| (6.83)|
|                    | 4 512     | 14.44| 17.50  | 11.82|
|                    |           | (9.81)| (11.70)| (6.60)|
|                    | 12 256    | 84.03| 85.02  | 83.32|
|                    |           | (48.18)| (45.27)| (72.35)|
|                    | 12 512    | 83.58| 84.28  | 83.28|
|                    |           | (47.64)| (45.44)| (72.80)|
| Transformer        | 4 256     | 16.32| 22.77  | 15.39|
|                    |           | (11.31)| (15.24)| (9.11)|
|                    | 4 512     | 15.91| 22.28  | 15.23|
|                    |           | (10.93)| (14.73)| (8.94)|
|                    | 12 256    | 16.79| 24.67  | 14.16|
|                    |           | (11.58)| (15.97)| (8.69)|
|                    | 12 512    | 17.45| 23.97  | 14.62|
|                    |           | (11.98)| (16.11)| (8.95)|
| SA-NCD             | 4 256     | 7.83 | 8.73   | 7.99 |
|                    |           | (5.97 )| (6.60 )| (4.66)|
|                    | 4 512     | 7.68 | 8.23   | 7.41 |
|                    |           | (5.75 )| (6.20 )| (4.28)|
|                    | 12 256    | 7.79 | 8.65   | 7.79 |
|                    |           | (5.84 )| (6.63 )| (4.48)|
|                    | 12 512    | 5.51 | 7.71   | 6.72 |
|                    |           | (4.82 )| (6.12 )| (4.00)|

Table 4: (Ablation study) Decoding performance of SA-NCD with and without pre-training step.

| Metric     | MacK | Common | Test |
|------------|------|--------|------|
| SA-NCD CER | 79.89| 79.19  | 81.63|
| w/o pre-training G | 83.36| 82.85  | 90.06|
| SA-NCD CER | 6.04 | 8.78   | 7.47 |
| w/o pre-training S | 5.19 | 7.38   | 4.75 |
| SA-NCD CER | 5.51 | 7.17   | 6.72 |
| WER         | 4.82 | 6.12   | 4.00 |

Figure 5: Pixel-wise prediction for current input \(t_n\) when typing each phrase. The letter inside ‘ ’ indicates the label of the current input, \(c_n\). Each bold character is marked at the average position.

Ablation study
For in-depth analysis, we conducted an ablation study of our SA-NCD network. Based on the best performing \((L, d_h) = (12, 512)\) configuration of SA-NCD, we explored how SA-NCD is affected without pre-training geometric decoder \((G)\), and pre-training semantic decoder \((S)\) (see Table 4).

SA-NCD w/o pre-training \(G\) suffered significantly compared to SA-NCD w/o pre-training \(S\). SA-NCD w/o pre-training \(G\) experienced unstable learning due to the characteristics of our input. This suggests that pre-training geometric decoder plays a key role in our training strategy because input \(T\) itself contains only geometric information. Therefore the semantic decoder converged well because a stable input \(G(X; \phi_G)\) with a fixed character index was provided by the geometric decoder. On the other hand, the decoding performance of SA-NCD improved slightly by pre-training \(S\).

Qualitative Visualization Results
We visualized the outcome of the decoder at a pixel level. When the current touch input \(t_n\) comes in given the previously touched input sequence \(\{t_1, \ldots, t_{n-1}\}\), the decoder outputs the character sequence \(\hat{c}_1, \ldots, \hat{c}_n\). We plotted every pair \((t_n, \hat{c}_n)\) to see how each predicted character by decoder is distributed over the typing area (see Figure 5). The area corresponding to the label is located widely by the decoder.

6 User Study
In order to examine the usability of IMK in real-life settings, we conducted a verification experiment. The details of the user study are described in the supplementary material, including the example video.

We measured WPM (words per minute) as a metric of the typing speed of each participant, where WPM is defined as...
In this work, we proposed a novel Invisible Mobile Keyboard (IMK) decoding task that eliminated the limitations of soft keyboards while maximizing the utility of mobile devices. We collected a suitable benchmark dataset for the task and designed a baseline approach, Self-Attention Neural Character Decoder (SA-NCD), with a new deep neural network structure and training scheme. SA-NCD led to significantly lower decoding error rates than the state-of-the-art methods of other sequential tasks. Besides, through a user study, we witness a typing speed of 51.6 WPM when using our proposed decoding algorithm, which surpassed all of the existing text-entry methods. Simultaneously, the users felt low physical, mental, and temporal demand when using our IMK with high satisfaction. The future works shall be on a front-end design for the practical implementation of IMK.

### References

[Al-Rfou et al., 2019] Rami Al-Rfou, Dokook Choe, Noah Constant, Mandy Guo, and Liion Jones. Character-level language modeling with deeper self-attention. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 33, pages 3159–3166, 2019.

[Chelba et al., 2014] Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Philipp Koehn, and Tony Robinson. One billion word benchmark for measuring progress in statistical language modeling. In Fifteenth Annual Conference of the International Speech Communication Association, 2014.

[Chung et al., 2014] Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. In NIPS 2014 Workshop on Deep Learning, December 2014, 2014.

[Costagliola et al., 2011] Gennaro Costagliola, Vittorio Fucella, and Michele Di Capua. Text entry with keyscratch. In Proceedings of the 16th international conference on Intelligent user interfaces, pages 277–286, 2011.

[Devlin et al., 2019] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 4171–4186, 2019.

[EnglishSpeak, 2019] EnglishSpeak. 1,000 most common english phrases. https://www.englishspeak.com/en/english-phrases, 2019. Accessed: 2019-09-30.

[Findlater and Wobbrock, 2012] Leah Findlater and Jacob Wobbrock. Personalized input: improving ten-finger touchscreen typing through automatic adaptation. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pages 815–824, 2012.

[Ghosh and Kristensson, 2017] Shaona Ghosh and Per Ola Kristensson. Neural networks for text correction and completion in keyboard decoding. arXiv preprint arXiv:1709.06429, 2017.

[Kim et al., 2019] Ue-Hwan Kim, Sahng-Min Yoo, and Jong-Hwan Kim. I-keyboard: Fully imaginary keyboard on touch devices empowered by deep neural decoder. IEEE Transactions on Cybernetics, 2019.

[Lu et al., 2017] Yiqin Lu, Chun Yu, Xin Yi, Yuanchun Shi, and Shengdong Zhao. Blindtype: Eyes-free text entry on handheld touchpad by leveraging thumb’s muscle memory. Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies, 1(2):1–24, 2017.
[MacKenzie and Soukoreff, 2003] I Scott MacKenzie and R William Soukoreff. Phrase sets for evaluating text entry techniques. In CHI’03 extended abstracts on Human factors in computing systems, pages 754–755, 2003.

[Shi et al., 2018] Weinan Shi, Chun Yu, Xin Yi, Zhen Li, and Yuanchun Shi. Toast: Ten-finger eyes-free typing on touchable surfaces. Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies, 2(1):1–23, 2018.

[Vaswani et al., 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Advances in neural information processing systems, pages 5998–6008, 2017.

[Vertanen et al., 2015] Keith Vertanen, Haythem Memmi, Justin Emge, Shyam Reyal, and Per Ola Kristensson. Velocitap: Investigating fast mobile text entry using sentence-based decoding of touchscreen keyboard input. In Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems, pages 659–668, 2015.

[Xiong et al., 2020] Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tieyan Liu. On layer normalization in the transformer architecture. In International Conference on Machine Learning, pages 10524–10533. PMLR, 2020.

[Zhu et al., 2018] Suwen Zhu, Tianyao Luo, Xiaojun Bi, and Shumin Zhai. Typing on an invisible keyboard. In Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems, pages 1–13, 2018.