KnuckleTouch: Enabling Knuckle Gestures on Capacitive Touchscreens using Deep Learning

Robin Schweigert
University of Stuttgart
Stuttgart, Germany
st112866@stud.uni-stuttgart.de

Jan Leusmann
University of Stuttgart
Stuttgart, Germany
jan.leusmann29@gmai.com

Simon Hagenmayer
University of Stuttgart
Stuttgart, Germany
st107469@stud.uni-stuttgart.de

Maximilian Weiß
University of Stuttgart
Stuttgart, Germany
maxuswhite@gmail.com

Huy Viet Le
University of Stuttgart
Stuttgart, Germany
mail@huyle.de

Sven Mayer
University of Stuttgart and
Carnegie Mellon University
info@sven-mayer.com

Andreas Bulling
University of Stuttgart
Stuttgart, Germany
andreas.bulling@vis.uni-stuttgart.de

ABSTRACT

While mobile devices have become essential for social communication and have paved the way for work on the go, their interactive capabilities are still limited to simple touch input. A promising enhancement for touch interaction is knuckle input but recognizing knuckle gestures robustly and accurately remains challenging. We present a method to differentiate between 17 finger and knuckle gestures based on a long short-term memory (LSTM) machine learning model. Furthermore, we introduce an open source approach that is ready-to-deploy on commodity touch-based devices. The model was trained on a new dataset that we collected in a mobile interaction study with 18 participants. We show that our method can achieve an accuracy of 86.8% on recognizing one of the 17 gestures and an accuracy of 94.6% to differentiate between finger and knuckle. In our evaluation study, we validate our models and found that the LSTM gestures recognizing archived an accuracy of 88.6%. We show that KnuckleTouch can be used to improve the input expressiveness and to provide shortcuts to frequently used functions.

CCS CONCEPTS

• **Human-centered computing** → **Touch screens**; Empirical studies in HCI; • **Hardware** → *Touch screens.*

KEYWORDS

KnuckleTouch, knuckle, input, DNN, CNN, LSTM

ACM Reference Format:
Robin Schweigert, Jan Leusmann, Simon Hagenmayer, Maximilian Weiß, Huy Viet Le, Sven Mayer, and Andreas Bulling. 2019. KnuckleTouch: Enabling Knuckle Gestures on Capacitive Touchscreens using Deep Learning. In *Mensch und Computer 2019 (MuC '19), September 8–11, 2019, Hamburg, Germany*. ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3340764.3340767

MuC '19, September 8–11, 2019, Hamburg, Germany

ACM ISBN 978-1-4503-7198-8/19/09…$15.00
https://doi.org/10.1145/3340764.3340767

1 INTRODUCTION

Over the last years, mobile devices evolved from being an additional device that people carry around to a primary computing and communication device. Nowadays, people perform a wide range of tasks on mobile devices, such as taking pictures, navigation, and connecting with friends. However, more complex tasks such as text editing are still mainly performed on desktop or laptop computers. A likely reason for this is that the input expressiveness of touch interaction is still limited.

While the mouse and keyboard traditionally gave a large variety to interact with the graphical user interface (GUI), today's touch interaction is mostly limited to a simple 2D touch coordinate on the screen. Holz and Baudisch [12] showed that touch is multidimensional but this is ignored by today's touch controllers. Therefore, researchers as well as touch device manufactures are investigating new input dimensions. For instance, recent iPhones can sense the pressure of the finger on the device (so-called ForceTouch). Harrison et al. [10] proposed to use the knuckle as an alternative input to enable an enriched interaction for touch surfaces. In their implementation, they used sound classification to identify knuckle input. Today, Huawei smartphones do use knuckle input for shortcuts such as screenshots but they offer no open source solution, which makes it difficult to study and further develop this input method on other mobile devices.

We present two ready-to-deploy models to build enhanced KnuckleTouch touch interfaces. First, we present a model to detect single knuckle inputs. Second, we present a long short term memory (LSTM) gesture recognizer with 17 trained gestures. We trained both models on ground truth capacitive images recorded from 18 participants and validated them in a second study with 12 participants. Our results show that our LSTM gesture recognizer achieves a 88.6% accuracy on the validation dataset and the Knuckle detector an accuracy of 94.6%.

The contribution of this paper is three-fold: 1) a dataset for training and testing containing 6.120 gestures (618.012 capacitive images) for finger and knuckle gestures and a separately collected validation dataset with 3.060 gestures; 2) ready-to-deploy models enabling KnuckleTouch and 17 gestures; and 3) an evaluation of KnuckleTouch gestures and a set of use cases.

2 RELATED WORK

In the following, we presented related work which leads towards knuckle input for commodity touch devices. 1) we present related work in the knuckle input domain. 2) we present work in conjunction with gesture recognizer. 3) we present work in which the raw capacitive sensor values are used to enhance touch interaction.

Knuckle input

For knuckle input a wide range of use cases have been proposed. The most prominent work regarding knuckle interaction is by Harrison et al. [10]. They propose that four different parts of the, finger can be differentiated: The finger pad, the fingertip, the nail, and the knuckle. Lopes et al. [21] use different hand gestures for actions such as copying, pasting and deleting objects on a tabletop. Qeexo[1] proposed "FingerSense". With FingerSense it is possible to differentiate between finger, knuckle, nail, and stylus input on a touchscreen. They propose a wide range of possible use cases such as an eraser tool for drawing or shortcuts. Finally, some new Huawei phones e.g. the Huawei P20 Pro offers shortcuts using the knuckle input such as a double tab for a screenshot.

In previous work on detecting the knuckle used various approaches. Harrison et al. [10] identify the different inputs based on changes in the acoustical spectrogram retrieved from conventional medical stethoscope with an electret microphone. With their system, they could differentiate between these four input methods with an accuracy of 95%. In contrast, Lopes et al. [21] use the sound of the gesture for input identification. They used the characteristics of the amplitude envelope and the fundamental frequency to detect different interactions. Chen et al. [5] enabled Knuckle input by using a smartwatch on the interacting arm to differentiate between tab and knuckle input.

Gesture Input

To detect user-defined gestures, previous work presented different recognizers. Rubine et al. [23] proposed the Gesture Recognizers Automated in a Novel Direct Manipulation Architecture (GRANDMA), a first attempt to add gesture interaction capability to direct manipulation interfaces. One year later, the same author presented a set of features to automatically recognize gestures [24], including angle, length and rotation features. Long et al. [20] extended the set of features by curviness and the aspect.

To reduce the effort when integrating gesture recognition into prototypes, Wobbrock et al. [1, 27, 28] presented three algorithms to recognize gestures using on an instance-based nearest-neighbor classifier with a Euclidean scoring function. Additionally. Li et al. [19] showed that his proposed Protractor showed advantages over both Rubine [23] and the DTW recognizers [30].

Today, more advanced gesture recognizer use machine learning (ML). Here, Gillian and Paradiso [7] presented the Gesture Recognition Toolkit (GRT) which is a cross-platform machine-learning library for real-time gesture recognition. Ten et al. [25] presented a different approach that uses multi-dimensional Dynamic Time Warping (DTW) to detect gestures. Recently, Google published QuickDraw[2] which uses neural networks to recognize user-drawn images. Trigueiros et al. [26] further compared four well-known machine learning algorithms to detect hand gestures and found that neural networks had a good performance.

Capacitive Recognition

A large body of work is dedicated to extracting information from capacitive sensors. The basic form is of data extraction is built-in in today's touch devices, where the touch controller performs a fast and straightforward extraction to determine the x and y position of the finger to perform an input action. However, Holz et al. [12] showed that there is more to the input than the center of the finger touching the sensor. Kumar et al. [14] presented an improved pipeline to extract the x and y position. In their implementation they used a Convolutional Neural Network (CNN) model to predict the position and gained a by 23.0% increases the touch accuracy.

Le et al. [18] presented a feasibility analysis of how to perform finger identification based on capacitive touchscreens using CNNs. Based on the possibility of a Fully Touch Sensitive Smartphones, Le et al. [17] presented a model which

[1] https://www.qeexo.com/ – last accessed 2019-04-04

[2] quickdraw.withgoogle.com/

can detect which finger is touching the one at which location. Holz et al. [13] presented a method to use the capacitive image data for continues authentification. Le et al. [15] proposed a neural network for detecting if a touchscreen input was done with a finger or the palm of the hand. The network was trained by taking capacitive images directly from the smartphone and then inserting them into a CNN. They reached an accuracy of 99.53%. Beyond modifier keys, one prominent scenario is to extract the finger orientation. Here, Xiao et al. [29] showed the first implementation using Gaussian process (GP). Followed by Mayer et al. [22] as they presented a CNN solution.

Summary

While the single components have been studied in the past, in this paper, we combine the three domains to foster a ready-to-deploy gesture recognizer for finger and knuckle input, which allows other researchers to build their system which supports knuckle input quickly.

3 CONCEPT AND USE CASES

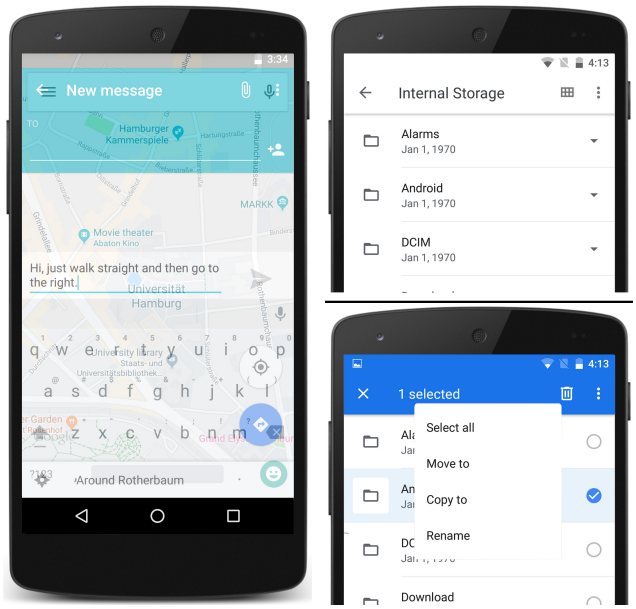
Previous work focused on identifying touches of knuckles while commercial devices use the identification for launching simple actions (e.g., taking screenshots). In the following, we describe four use cases to demonstrate how knuckle recognition and knuckle gestures can be used improve mobile touch interaction.

Improving Multitasking

Previous work [8, 18] presented the concept of porous user interfaces in the context of finger-aware interaction. The authors proposed that the GUI displays two different semi-transparent interfaces and dedicated fingers can only interact with one of the two interfaces. Similarly, we propose to use the differentiation of finger and knuckle touches to interact with two overlaying applications as shown in Figure 1a. Thereby, the finger interacts with the application in the foreground while the knuckle interacts with the background application. For example, users could type in a messaging application with the finger while browsing for information (e.g., looking for a location to tell a friend) could be done with the knuckle.

Multiple Input Modes

KnuckleTouch can also be used as a different input dimension, similar to how a computer mouse has at least two buttons to activate different functions at the same cursor position. For example, users can draw in a painting application with the finger tip while the knuckle can be used to erase the drawn content. This concept can also be applied to other application domains. The finger tip could be responsible for the main function while the knuckle can be used to perform



(a) (b)

Figure 1: Figure (a) shows a porous interface for improving multitasking where users can navigate the map in the background with the knuckle while typing with the finger comfortably in the messenger app. Figure (b) shows the state change when interacting with a KnuckleTouch shortcut, first the user only gets presented the list view and by using a KnuckleTouch directly opens the pop-up menu.

secondary actions such as opening a context menu (see Figure 1b). Previous work presented GUI widgets that could also be used with our model [4].

Text Editing Shortcuts

Mobile text editing is inconvenient due to the lack of shortcuts and the occlusion caused by the fat-finger problem. We propose using *KnuckleTouch* to provide shortcuts related to text editing. Instead of selecting text with direct touch, a swipe gesture to the left or right with the knuckle could be used which is independent from text size and occlusion. Moreover, since copy and paste operations are only accessible with long-presses in recent user interfaces (UIs) nowadays, respective knuckle gestures could be used to avoid inconvenient dwell times.

In general, knuckle gestures can be used to provide shortcuts to frequently used functions similar to how modifier keys on hardware keyboards are used to access functions such as copy, paste, and text selection.

Table 1: Gestures and applications defined by our gesture survey.

No.	Gesture	Example application
1	Tap	Right click
2	Two Tap	Multitask view
3	Swipe left	Rotate 3D object
4	Swipe right	
5	Swipe up	
6	Swipe down	
7	Two swipe up	Open phone setting
8	Two swipe down	Open app settings
9	Circle	Open camera
10	Arrowhead left	Switch between apps
11	Arrowhead right	
12	✓	Confirm input
13	Γ	Toggle Flashlight
14	L	Open custom app
15	⅃	
16	S	Screenshot
17	Rotate	Control volume

Browsing and Navigation Shortcuts

Retrieving information on a desktop or laptop computer (e.g., browsing the internet) often requires switching between tabs and applications, copying and pasting content, and sharing content with other contacts. *KnuckleTouch* could be used to provide convenient shortcuts for these actions. We envision app switching using a knuckle arrowheads gesture while copying and pasting content could be performed similar to what we proposed above.

4 DATA COLLECTION STUDY

To gather data, we follow the research approach pipeline presented by Le et al. [18] to train Deep Neural Network (DNN) in the context of human-computer interaction (HCI). As first step, we conducted a user study to collect labeled touch data while participants performed touch and knuckle input on a touch device. To understand not only the simple input but also gesture input, we run an initial gesture elicitation survey to collect possible knuckle gestures. In the data collection study, we asked participants to perform these gestures.

Gesture Elicitation Survey

We asked 11 participants (all male, age 22-29) if they can imagine using knuckle as an input for certain actions. We presented various unistroke gestures which all had the potential to be performed with the knuckle and the finger to participants to better understand how they are perceived by the participant. Thus, in the survey we let participants comment on a set of gestures and how they could be mapped to the different example applications, see Table 1âĹĘâĹĘâĹĘ.

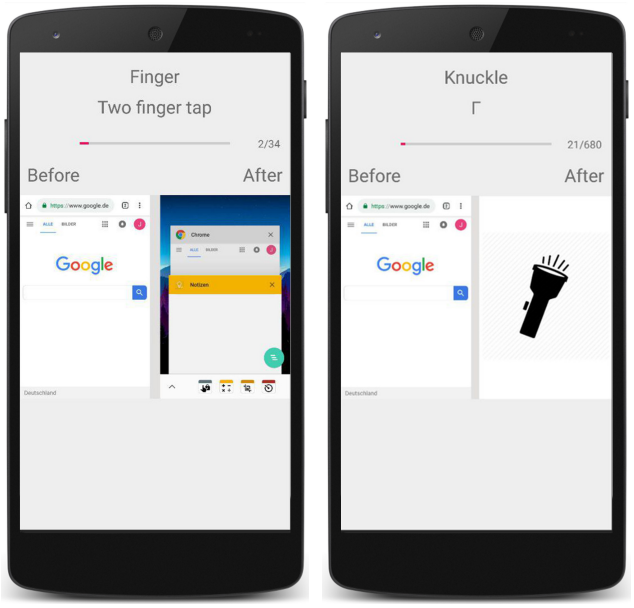
We found that participants did overall not favor two knuckle gestures as they are too cumbersome to perform. However, as they pose an interesting addition to simple knuckle input, we decided to keep three gestures in the final gesture set. Moreover, all selected gestures can also be performed with a finger. By using the most popular gestures, we, in the following, use the gestures presented in Table 1.

Apparatus

We used two devices, an Android LG Nexus 5 retrieves the capacitive images, and a laptop running a Python script to control the application running on the Nexus 5. We used an LG Nexus 5 running Android 5.1.1 with a modified kernel to access the 27×15 8-bit raw capacitive images of the Synaptics ClearPad 3350 touch sensor. The modified kernel was set up to capture a capacitive image every 50ms, c.f. Le et al. [16]. The Python script enabled the experimenter to walk the participants through the whole study. Moreover, the Python script presented a live visualization of the import to the experimenter. In case the gesture was not correctly captured or performed, the experimenter was able to initiate to redo the last input. Moreover, the participants had the chance to repeat the gesture by notifying the experimenter to repeat the gesture. The Android application showed the task to the participants. Participants were displayed the next gesture to perform, as well as the example tasks as described in Table 1.

Procedure

We conducted the study in a laboratory environment to have as little distraction for the participants as possible. After welcoming the participants, they were introduced to the study here, participants were told that they could take a break or quit the study at any point. Afterward, we asked them to fill a consent form and a demographics questionnaire. First, participants performed a tutorial where they had to perform each of the 17 gestures once with the finger and once with the knuckle. All gestures were performed with the right hand. Here the experimenter explained each use case for the gestures. The experimenter explained that the example use cases were only meant for the usage with the knuckle, but for data collection purposes they would also have to do them with the finger as well. Then the data collection began. All of



(a) (b)

Figure 2: Screenshots of the data collection app.

the 17 gestures had to be executed 10 times with their finger and knuckle each, resulting in $17 \times 10 \times 2 = 340$ samples. Participants either first performed all knuckle gestures and then all finger gestures and vice versa, based on a Latin square balanced. The gestures within the two blocks where randomized.

Participants

We recruited participants from an internal university self-volunteer pool. The 18 volunteer participants (4 female and 14 male) were between 21 and 26 years old ($M = 24.2$, $SD = 1.4$). The duration of the whole study was approximately 45 minutes. No participant had any movement impairments. Every participants' dominant hand was their right hand.

5 MODELLING KNUCKLETOUCH

In the following, we describe two detection mechanisms. First, we describe a model which enables to classify finger vs. knuckle input. Second, we present a model to recognize gestures performed by fingers or knuckles.

Dataset and Preprocessing

In our data collection study, we collected 141.310 capacitive images with a finger or knuckle present during the tasks from 6.120 gestures samples from 18 participants.

Knuckle vs. Finger Classifier

To build a classifier to differentiate between finger and knuckle input, we first transformed the dataset to contain only single finger or knuckle inputs. Therefore, we extracted each touch from the recorded time series dataset and used each touch as a sample. For this task, we used the blob detection available via OpenCV v4.0. From the gestures, we extracted 154, 503 unique blobs. The average blob size of a finger input is $16.6px^2$ ($SD = 5.3px^2$) and for a knuckle $13.7px^2$ ($SD = 3.7px^2$). Furthermore, we augmented our dataset by flipping the recognized blobs once horizontally, once vertically, and once in both direction. This resulted in a augmented dataset containing 503, 886 capacitive images, examples are shown in Figure 3. Finally, we pasted the blobs into the upper left corner of an empty 27×15 image, c.f. Mayer et al. [22].

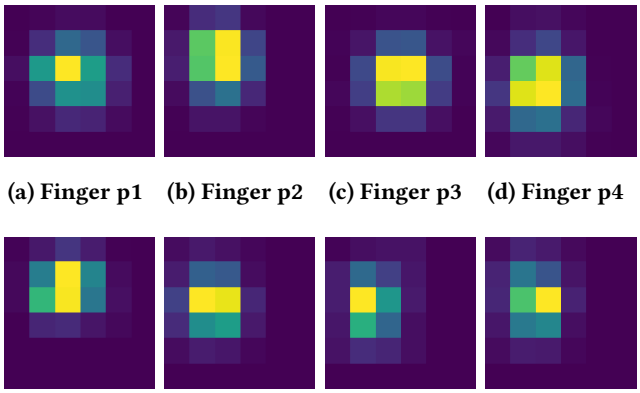
Baseline. First, we determined a baseline recognition rate by using well-established ML models to classify finger vs. knuckle input. We first extracted the following features: the sum of capacitance, avg of capacitance ellipse area, ellipse width, ellipse height, and ellipse theta. We used the same features for the baseline as Le et al. [15]. We used ZeroR, DT, RF, *k*NN, and SVM. We performed a grid search with a 5-fold approach to determine the most suitable hyperparameters for all four models. The results can be seen in Table 2. We used a 13 : 5 participants wise split for the train and test set. Here, we found a maximum accuracy of 79% for the test set when using a RF model.

Deep Neural Network. To improve upon the baseline models, we implemented a CNN using *Keras* (based on the TensorFlow backend). CNNs are specially tuned to learn and represent image data like the capacitive image. We used the trial-and-error method [6] combined with a grid search to find parameters for the CNN model. Again, using the 13 : 5 participants wise split for the train and test set.

We determined that the deep CNN with four convolution layer and two dense layers yield the best accuracy. The model

Table 2: Results of the knuckle vs. finger classifier. Here, we present the results of basic machine learning algorithms as well as the best CNN model of the test set.

	Param	Prec.	Rec.	Acc.
ZeroR		.5	.5	.5
*k*NN	$k = 2$	.73	.71	.71
SVM	$C = 10$ & $gamma = 10.$	.72	.72	.72
DT	$max\ depth = 22$	.67	.73	.73
RF	$n = 63$ & $max\ depth = 60$	.79	.79	.79
CNN	4 conv x 2 dense	.95	.92	.95



(a) Finger p1 (b) Finger p2 (c) Finger p3 (d) Finger p4

(e) Knuckle p1 (f) Knuckle p2 (g) Knuckle p3 (h) Knuckle p4

Figure 3: The top row shows four random samples of finger input form different participants. The bottom row shows for knuckle inputs from the corresponding participants.

structure is depicted in Figure 4. The input blob image fed to four convolution layer one after the other. After every second convolution layer a max-pooling layer is applied. The last convolution layer is followed by two dense layers with 140 and 70 neurons. Additionally, dropout is applied after every convolution layer with 45%, and in front of the dense layers with 50%. Also, every layer has an L1L2-regularization with a factor of (0.005, 0.015) to counteract possible overfitting. Bach-normalization is also applied after every convolutional layer. The output layer has 2 neurons with a softmax activation function. If we do not report a hyperparameter, we used the standard value (e.g., optimizer settings) as reported in Keras' documentation.

As loss function, we used the cross-entropy loss. The training ran with an Adam optimizer with a learning rate starting from .001 with a reduction by 5% after 140 epochs without improvement and a minimal learning rate of .00001. We used a batch size of 2000. We used an early stopping approach. Our Network performed best after 33,767 epochs. The model was training for approximately 7 days on an Nvidia Tesla V100. Thus, our final model achieves an accuracy of 95.8% (the recall is 94.3%, and the precision is 95.7%) on the train set. On the test set the accuracy is 94.6%, the recall is 92.7%, and the precision is 95.6%.

Finally, as a gesture contains multiple raw capacitive images, we build an ensemble to classify if the whole gesture was performed with a finger or knuckle. Therefore, we predict the class per input same of a gesture and choose the predicted class based on class frequency. Our ensemble achieved a 98.3% accuracy on the train set. The accuracy is 97.3% on the test set (the recall is 97.9%, and the precision is 96.7%).

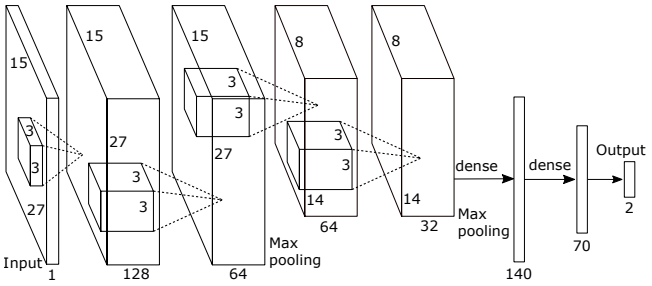


Figure 4: An illustration of the architecture of our CNN model which we used to differentiate between knuckle and finger.

Neural Network for Gesture Recognition

We used a CNN-LSTM to train a gesture recognizer. Again we used the raw capacitive images as input to predict one of the 17 gestures, see Table 1. A gesture was on average 15.9 images ($SD = 13.6$, $min = 1$, $max = 301$). We used the zero padding approach to feed the raw capacitive images into the LSTM. Thus, gestures which are too long will not feed completely to the network and gestures which are to short are getting filled zeros. We found feeding 50 images is sufficient for the model to predict the gesture. Here, only 3% of all gestures got cut off. Again we used the trial-and-error method [6] combined with a grid search for hyper-parameters tuning using the 13 : 5 participants wise split for the train and test set.

We determined that the CNN-LSTM with four convolution layer and LSTM layer yield the best accuracy. The model structure is depicted in Figure 5. 50 capacitive imaged are feed into the in input layer to determine the gesture. The input is passed on to `TimeDistributed`[3] CNN layers to them to LSTM layers. We found that four-time distributed convolution layers followed by two LSTM layers worked the best. The convolution layers all used a 3×3 kernel with padding set to be same and the filter count is 64 on the upper layer, followed by 32, 32 and 16 filters. After every second convolution layer, a max-pooling layer is added with a kernel size of 2×2. Additionally, dropout is applied after every convolution layer with 50%, and 25% after each LSTM layer. The LSTM layer had 80 and 50 neurons. Also, every layer has an L1L2-regularization with a factor of (0.005, 0.015) to counteract possible overfitting. The output layer has 17 neurons with a softmax activation function. If we do not report a hyperparameter, we used the standard value (e.g., optimizer settings) as reported in Keras' documentation.

We found that an Adam optimizer with a learning rate of 0.0001 and a batch size of 50 works best. The model was

[3]TimeDistributed layer in Keras: https://keras.io/layers/wrappers/#TimeDistributed

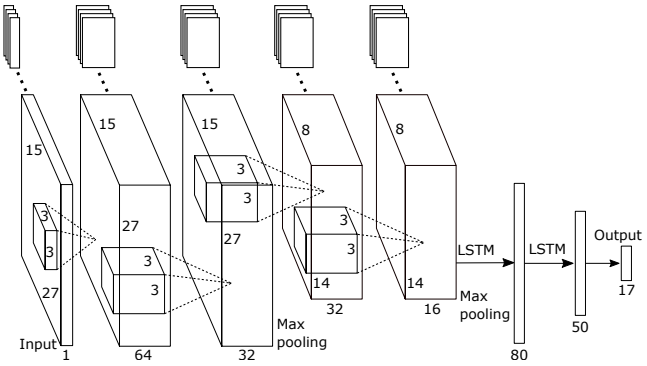


Figure 5: An illustration of the architecture of our CNN-LSTM model which we used to classify gestures.

trained for 3000 epochs. We reached an accuracy of 90.6% on the training set and 84.4 on the test set. To further boost the performance, we performed a warm start and retrained the network with the exact same setting. Here, we used an early stopping approach. Our Network performed best after 398 epochs. Thus, or final model achieves 97.9% on the train set and 86.8% on the test set.

Mobile Implementation

We froze these models and used TensorFlow Lite[4] for Android to run them directly on the Nexus 5. We implemented the preprocessing pipeline in JAVA using OpenCV for Android. The CNN can detect each input real time, the average prediction time is $42.ms$ ($SD = 9.7$). The CNN-LSTM waits for 50 images, or a touch-up event to feed the data into the network with the zero padding approach. The prediction time is $1066.8ms$ ($SD = 77.2$).

6 KNUCKLETOUCH EVALUATION

To evaluate KnuckleTouch, we conduct a second study with two parts. In the first part, we collect a validation dataset to evaluate the model quality of the CNN and the LSTM. In the second, we get to use the gestures in a text editing scenario.

Apparatus

For the first part, we used the exact hardware and implementation of our data collection study. For the second part, we implemented a text selection, copy, past, and app switching task using gestures as well as the standard Android UI. While not all gestures were used in the next task, the full CNN-LSTM model was used which predicts the 17 gestures.

Procedure

We conducted the study in a laboratory environment to have as little distraction for the participants as possible. After welcoming the participants, they were introduced to the study here. Participants were told that they could take a break or quit the study at any point. Afterward, we asked them to fill a consent form and a demographics questionnaire.

The first part of the study was exactly as in the first study, participants performed a tutorial. Afterward, all 17 gestures had to be performed 15 times with their finger and knuckle each to collate validation data.

In the second part, participants performed a text editing task using the gestures, and the standard Android UI. The order was counterbalanced. In each condition, participants were asked to select three words and copy-and-past them into a second app. Word selection as implemented by the swipe left and right knuckle gesture, copy with the two tap knuckle gesture, the knuckle arrowheads are the app switching gestures, and the knuckle circle would past data. After each scenario participants were asked to fill in a raw NASA-Task Load Index (raw TLX), and a system usability scale (SUS). Inspired by Le et al. [17], we further collected qualitative feedback about the perceived easiness, speed, success, accuracy, and comfort on a 7-point Likert scale.

Finally, we interviewed to understand how the participants overall liked the idea of KnuckleTouch. In total, this part took around 15 minutes per participant.

Participants

We recruited 12 participants who did not participate in the data collection study. The 12 participants (5 female and 7 male) were between 21 and 32 years old ($M = 25.3, SD = 4.5$). The duration of the whole study was approximately 1 hour. No participant had any movement impairments, and every participants' dominant hand was their right hand.

7 RESULTS

To validate our models, we conducted a second study with 12 participants in which we recorded an additional 3,060 gestures.

Model Accuracy

We mainly conducted the second study to collect new capacitive images from participants which have not taken part in the first study. This is crucial to collect a model validation set. In the follwoing we used the new validation set to validate both the e CNN to differentiate between finger and knuckle as well as to validate the CNN-LSTM gesture recognizer. The validation set consist our of 3,060 gestures obtained in the evaluation study. The total number of capacitive images is 618,012.

For the CNN-LSTM gesture recognizer (see Figure 5) we found that gestures were recognized with an accuracy of 88.6% with a recall of 88.5% and an F1-score of 88.6%. The accuracy of the validation set is even a bit better then the

[4]https://www.tensorflow.org/lite

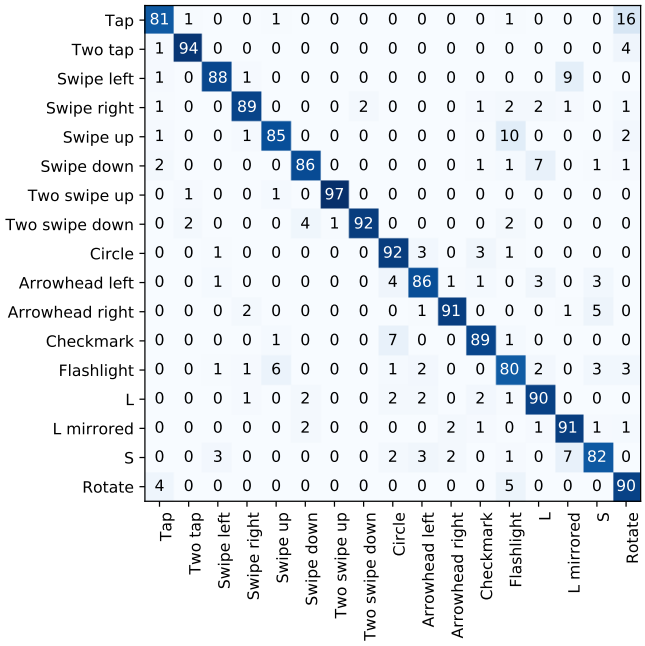


Figure 6: Confusion matrix of the CNN-LSTM for identifying 17 gestures with an accuracy of 93.2%**. The values in the figure represent the classification results in percent (%). The x-axis represent the predicted class and the y-axis the actual class.**

smaller test set with only 5 participants. This confirms that our model generalized and is not prawn to overfitting.

The CNN to differentiate between finger and knuckle from a single capacitive image (see Figure 4) achieved an accuracy of 93.2% on the validation set, see Figure 6. The recall was 90.9% and the precision 93.2%. Moreover, the final CNN ensemble achieved an accuracy of 97.1%, the recall is 96.9%, and the precision is 95.6%. The accuracy values for both the CNN as well as the CNN ensemble are in line with the reported 94.6% and 97.3% arrays values on the test set. Thus, the model is stable and does not overfit.

Qualitative Feedback in Realistic Scenarios

In the realistic scenario of the evaluation study we asked participants to fill in a SUS questionnaire [3], raw TLX [11], as well as the subjective perception by Le et al. [15]. The qualitative results overall showed that finger input was easier to perform then the knuckle input, see Table 3. However, participants reported that they used KuckleTouch already 255 times in the first part of the study which as they reported reduced their enthusiasm due to the extensive use before.

Interview Feedback

We conducted interviews with 12 participants. We combined all interviews from the sessions for analysis. We transcribed the interviews literally while not summarizing or transcribing phonetically [2]. Finally, we employed a simplified version of qualitative coding with affinity diagramming [9] for interview analysis.

We first asked partiapnts about their overall impression. The majority stated that is easy to perform and they were generally positive (P1, P3, P4, P5, P6, P7, P10, P12). P10 stated *"the gestures are intuitive, thus, easy to learn, and are offering many of possibilities [for new input]"*. They state for instance that KnuckleTouch is *"interesting"* (P4, P10), *"useful"* (P3, P7), *"practical"* (P7), and *"easy"* (P3). Moreover, P3 and P7 referred to the new interaction with the knuckle as *"impressive"*. On the other hand, participants (P1, P2, P4, P5, P9, P8, P10, P12) stated that extensive use of KnuckleTouch is unpleasant. Additionally, P6, P7, and P12 stated that gestures with two knuckles are harder to perform. Finally, they (P1, P4, P5, P6, P7, P8, P11) comments concerning the familiarity of KnuckleTouch. Thus, we argue that the input suffers from familiarity in the given scenario and the fact that participants were asked to perform the input 255 times if strongly reflected in their comments. Here, they also argued themselves that they see the input to be used not as often, here P10 stated: *"maybe 10-20 times a day"*. Additionally, P4 stated: *"when I use it regularly, it will be normal a interaction"*.

We additionally asked participants about the advantages and disadvantages to better understand why they would use KnuckleTouch or not. Four participants (P2, P5, P8, P11) did not see any advantages while only one would not try it at the same time. Five participants (P1, P4, P6, P9, P12) said that the advantages depend on the implementation, reliability, as well as the use case.

Table 3: Qualitative results from copy and paste task in both conditions. raw TLX on a 21-point scale (0-20), the SUS on its' standard scale from 0 to 100, and the subjective perceptions (7-point Likert scale) as described by Le et al. [15]. We used a Friedman test to determine statistically significant difference between Finger and Knuckle

	Finger		Knuckle			
	M	**SD**	**M**	**SD**	χ^2	**p-value**
raw TLX	2.0	1.6	6.6	2.5	12.	<.001
SUS	89.6	8.8	57.9	16.1	8.3	<.004
Easiness	6.7	0.8	4.8	1.6	8.3	<.004
Speed	6.7	0.5	2.9	1.2	12.	<.001
Success	6.1	1.6	3.6	1.7	6.4	<.011
Accuracy	6.2	1.3	3.4	1.4	11.	<.001
Comfort	6.7	0.7	3.1	1.4	12.	<.001

We asked participants if they would use KnuckleTouch if their next phone would offer this new feature. Here, we found that four participants (P8, P9, P11, P12) were reluctant. Two participants (P5, P6) were reluctant and stated that if there were no technical issues they would use it. Six participants would use KnuckleTouch (P1, P2, P3, P4, P7, P10).

Technical issues were brought up by five participants (P2, P3, P4, P7, P8). Here they mainly stated that the model is too slow and no feedback is presented during the model evaluation process. A comment by P6 is remarkable to understand the overall results of the interviews and questioners: *"if the prototype was to work more smoothly and faster, I might have a different impression on the interaction."* Participants saw also input issues related to the fat-finger problem, here three participants (P2, P5, P9) argued that KnuckleTouch is an imprecise input.

Three participants (P3, P5, P7) stated that they would envision KnuckleTouch to perform different functions and not replace the ones which are already easy to perform. *"actions which are normally super long-winded to reach"* - P7. Nine participants (P2, P3, P4, P6, P7, P8, P10) see KnuckleTouch as a possibility to input shortcuts, such as *"screenshots"* - P3, *"volume"* - P7, *"turn on alarm for the next morning"* - P2, and *"next or second next song"* - P10. They also saw KnuckleTouch as an extra input dimension, this was the most prominent use case among our participants (P2, P3, P4, P5, P6, P8, P9, P10, P12). On the one hand, they saw it as a system-wide feature, *"it is a new input, [...] an additional new input dimension"* - P5. On the other hand, they also could envision it as an app specify interaction, *"alternative stroke [in a painting app]"* - P4. Finally, P9 additionally stated that KnuckleTouch would be useful for other devices types, e.g., tabletops, and wall-sized displays.

8 DISCUSSION

Our CNN-LSTM gesture recognizer achieved an 88.6% accuracy in our validation study, and our CNN KnuckleTouch recognizer an accuracy of 93.2%. The results showed that both models are not prone to overfitting and thus, generalize well to new data.

In contrast to proprietary approaches such as the one from Huawei, or techniques that are based on the sound of impact [10], our approach works on virtually all recent smartphones with a mutual capacitive touchscreen. We further share this approach with the community to enable them to use KnuckleTouch and the gestures on commodity smartphones. As today's touch devices all have a capacitive resolution of around $4 \times 4mm$ per pixel, the models can easily be deployed on different devices. Deploying the models on more modern phones will also reduce prediction time as phones nowadays are equipped with a dedicated ML acceleration unit. Reducing the prediction time and, therefore, counteracting the comments by participants that the system is to slow, will likely also increase the overall rating of the current implementation.

Participants perceived our use cases, that we built upon the models, as easy to perform and intuitive. However, the extensive use lowered the results for the qualitative feedback as well as the participants' enthusiasm. Due to that, participants were not used to using knuckles as an input method. Using knuckles, e.g., for taking screenshots, is recently only implemented on Huawei devices. Thus, this study is a excessive test for KnuckleTouch as participants were asked to perform uncommonly many KnuckleTouch gestures. We propose a long-term study in which we are certain that the results will reveal more natural feedback.

Our evaluation study gave us the opportunity to ask participants for potential use cases when KnuckleTouch gestures could be used. Our interview analysis revealed two common themes to use the new input: 1) KnuckleTouch as an additional input dimension, and 2) KnuckleTouch as shortcuts. Also clear from the comments is that our participants do not want to substitute already available functions. They stated that they want additional functions or hard to access functions faster to reach using KnuckleTouch. This also contributes to the fact that they disliked the KnuckleTouch gestures in the realistic scenario as, for instance, app switching is already easy to perform and well-known.

In summary, our evaluation shows that KnuckleTouch can be used to provide shortcuts to frequently used functions and to improve the touch expressiveness. Further, potential users perceived KnuckleTouch as intuitive and easy to perform while it can be accurately recognized on off-the-shelf smartphones.

9 CONCLUSION

In this paper, we presented KnuckleTouch to enrich the expressiveness for touch-based devices. In a first step, we presented a CNN model to differentiates touches from fingers and knuckles based on single capacitive images of commodity smarpthones. Our model achieved an accuracy of 93.2% using a validation set. In the next step, we extend our model to recognize gestures using a CNN-LSTM with an accuracy of 88.6% . By using an ensemble of CNNs, we further show an accuracy of 97.1% can be achieved for differentiating between fingers and knuckles during gesture recognition.

In a subsequent evaluation study, we show that our model generalizes well. Our participants perceived KnuckleTouch as easy to learn, intuitive, practical, and offers many possibilities for new input. However, the study revealed that the extensive use of KnuckleTouch is unpleasant due to the unfamiliarity of this input method. Thus, future work could evaluate KnuckleTouch in a long-term study to consider this

effect. For this, we provide our dataset and the models which are ready-to-deploy on commodity touch-based devices. It is also left to future work to explore this input method for mobile situations, such as while walking and being encumbered.

10 DATASET AND MODEL

We publicly release the dataset and together with jupyter notebooks to run Python 3.6 code to process the data and train the models. All models are trained and tested with TensorFlow 1.13.1. Additionally, we provide ready to deploy models for both the CNN classifier and the CNN-LSTM gesture recognizer. This will enable other researchers to build upon the presented research. The code is available under MIT license here:
https://git.perceptualui.org/public-projects/knuckletouch

ACKNOWLEDGMENTS

This work was financially supported by the German Research Foundation (DFG) within Cluster of Excellence in Simulation Technology (EXC 310/2) at the University of Stuttgart. Moreover, this project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme under grant agreement No 801708.

REFERENCES

[1] Lisa Anthony and Jacob O. Wobbrock. 2010. A Lightweight Multistroke Recognizer for User Interface Prototypes. In *Proceedings of Graphics Interface 2010 (GI '10)*. Canadian Information Processing Society, Toronto, Ont., Canada, Canada, 245–252. http://dl.acm.org/citation.cfm?id=1839214.1839258

[2] Ann Blandford, Dominic Furniss, and Stephann Makri. 2016. *Qualitative Hci Research: Going Behind the Scenes*. Morgan & Claypool Publishers. 1–115 pages. https://doi.org/10.2200/S00706ED1V01Y201602HCI034

[3] John Brooke. 1996. SUS-A quick and dirty usability scale. *Usability evaluation in industry* 189, 194 (1996), 4–7.

[4] Daniel Buschek and Florian Alt. 2017. ProbUI: Generalising Touch Target Representations to Enable Declarative Gesture Definition for Probabilistic GUIs. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17)*. ACM, New York, NY, USA, 4640–4653. https://doi.org/10.1145/3025453.3025502

[5] Xiang 'Anthony' Chen, Tovi Grossman, Daniel J. Wigdor, and George Fitzmaurice. 2014. Duet: Exploring Joint Interactions on a Smart Phone and a Smart Watch. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '14)*. ACM, New York, NY, USA, 159–168. https://doi.org/10.1145/2556288.2556955

[6] Paulin Coulibaly, François Anctil, and Bernard Bobée. 2000. Daily reservoir inflow forecasting using artificial neural networks with stopped training approach. *Journal of Hydrology* 230, 3 (2000), 244 – 257. https://doi.org/10.1016/S0022-1694(00)00214-6

[7] Nicholas Gillian and Joseph A. Paradiso. 2014. The Gesture Recognition Toolkit. *J. Mach. Learn. Res.* 15, 1 (Jan. 2014), 3483–3487. http://dl.acm.org/citation.cfm?id=2627435.2697076

[8] Aakar Gupta, Muhammed Anwar, and Ravin Balakrishnan. 2016. Porous Interfaces for Small Screen Multitasking Using Finger Identification. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology (UIST '16)*. ACM, New York, NY, USA, 145–156. https://doi.org/10.1145/2984511.2984557

[9] Gunnar Harboe and Elaine M. Huang. 2015. Real-World Affinity Diagramming Practices: Bridging the Paper-Digital Gap. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI '15)*. ACM, New York, NY, USA, 95–104. https://doi.org/10.1145/2702123.2702561

[10] Chris Harrison, Julia Schwarz, and Scott E. Hudson. 2011. TapSense: Enhancing Finger Interaction on Touch Surfaces. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology (UIST '11)*. ACM, New York, NY, USA, 627–636. https://doi.org/10.1145/2047196.2047279

[11] Sandra G. Hart. 2006. NASA-Task Load Index (NASA-TLX); 20 years later. In *Proceedings of the Human Factors and Ergonomic Society annual meeting*, Vol. 50. SAGE Publications, SAGE Publications, Los Angeles, CA, USA, 904–908. https://doi.org/10.1177/154193120605000909

[12] Christian Holz and Patrick Baudisch. 2010. The Generalized Perceived Input Point Model and How to Double Touch Accuracy by Extracting Fingerprints. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '10)*. ACM, New York, NY, USA, 581–590. https://doi.org/10.1145/1753326.1753413

[13] Christian Holz, Senaka Buthpitiya, and Marius Knaust. 2015. Bodyprint: Biometric User Identification on Mobile Devices Using the Capacitive Touchscreen to Scan Body Parts. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI '15)*. ACM, New York, NY, USA, 3011–3014. https://doi.org/10.1145/2702123.2702518

[14] Abinaya Kumar, Aishwarya Radjesh, Sven Mayer, and Huy Viet Le. 2019. Improving the Input Accuracy of Touchscreens using Deep Learning. In *Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems* (2019-05-04) *(CHI'19 EA)*. ACM, New York, NY, USA. https://doi.org/10.1145/3290607.3312928

[15] Huy Viet Le, Thomas Kosch, Patrick Bader, Sven Mayer, and Niels Henze. 2018. PalmTouch: Using the Palm As an Additional Input Modality on Commodity Smartphones. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18)*. ACM, New York, NY, USA, Article 360, 13 pages. https://doi.org/10.1145/3173574.3173934

[16] Huy Viet Le, Sven Mayer, Patrick Bader, and Niels Henze. 2017. A Smartphone Prototype for Touch Interaction on the Whole Device Surface. In *Proceedings of the 19th International Conference on Human-Computer Interaction with Mobile Devices and Services (MobileHCI '17)*. ACM, New York, NY, USA, Article 100, 8 pages. https://doi.org/10.1145/3098279.3122143

[17] Huy Viet Le, Sven Mayer, and Niels Henze. 2018. InfiniTouch: Finger-Aware Interaction on Fully Touch Sensitive Smartphones. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology (UIST '18)*. ACM, New York, NY, USA, 779–792. https://doi.org/10.1145/3242587.3242605

[18] Huy Viet Le, Sven Mayer, and Niels Henze. 2019. Investigating the Feasibility of Finger Identification on Capacitive Touchscreens Using Deep Learning. In *Proceedings of the 24th International Conference on Intelligent User Interfaces (IUI '19)*. ACM, New York, NY, USA, 637–649. https://doi.org/10.1145/3301275.3302295

[19] Yang Li. 2010. Protractor: A Fast and Accurate Gesture Recognizer. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '10)*. ACM, New York, NY, USA, 2169–2172. https://doi.org/10.1145/1753326.1753654

[20] Allan Christian Long, Jr., James A. Landay, and Lawrence A. Rowe. 1999. Implications for a Gesture Design Tool. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '99)*. ACM, New York, NY, USA, 40–47. https://doi.org/10.1145/302979.302985

[21] Pedro Lopes, Ricardo Jota, and Joaquim A. Jorge. 2011. Augmenting Touch Interaction Through Acoustic Sensing. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces (ITS '11)*. ACM, New York, NY, USA, 53–56. https://doi.org/10.1145/2076354.2076364

[22] Sven Mayer, Huy Viet Le, and Niels Henze. 2017. Estimating the Finger Orientation on Capacitive Touchscreens Using Convolutional Neural Networks. In *Proceedings of the 2017 ACM International Conference on Interactive Surfaces and Spaces (ISS '17)*. ACM, New York, NY, USA, 220–229. https://doi.org/10.1145/3132272.3134130

[23] Dean Rubine. 1991. Specifying Gestures by Example. In *Proceedings of the 18th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '91)*. ACM, New York, NY, USA, 329–337. https://doi.org/10.1145/122718.122753

[24] Dean Harris Rubine. 1992. *The Automatic Recognition of Gestures*. Ph.D. Dissertation. Pittsburgh, PA, USA. UMI Order No. GAX92-16029.

[25] Gineke A. ten Holt, Marcel J. T. Reinders, and Emile A. Hendriks. 2007. Multi-dimensional dynamic time warping for gesture recognition. In *Thirteenth Annual Conference of the Advanced School for Computing and Imaging*, Vol. 300.

[26] Paulo Trigueiros, António Fernando Ribeiro, and L. P. Reis. 2012. A comparison of machine learning algorithms applied to hand gesture recognition. In *7th Iberian Conference on Information Systems and Technologies (CISTI'12)*. 1–6.

[27] Radu-Daniel Vatavu, Lisa Anthony, and Jacob O. Wobbrock. 2012. Gestures As Point Clouds: A $P Recognizer for User Interface Prototypes. In *Proceedings of the 14th ACM International Conference on Multimodal Interaction (ICMI '12)*. ACM, New York, NY, USA, 273–280. https://doi.org/10.1145/2388676.2388732

[28] Jacob O. Wobbrock, Andrew D. Wilson, and Yang Li. 2007. Gestures Without Libraries, Toolkits or Training: A $1 Recognizer for User Interface Prototypes. In *Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology (UIST '07)*. ACM, New York, NY, USA, 159–168. https://doi.org/10.1145/1294211.1294238

[29] Robert Xiao, Julia Schwarz, and Chris Harrison. 2015. Estimating 3D Finger Angle on Commodity Touchscreens. In *Proceedings of the 2015 International Conference on Interactive Tabletops & Surfaces (ITS '15)*. ACM, New York, NY, USA, 47–50. https://doi.org/10.1145/2817721.2817737

[30] Shumin Zhai and Per-Ola Kristensson. 2003. Shorthand Writing on Stylus Keyboard. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '03)*. ACM, New York, NY, USA, 97–104. https://doi.org/10.1145/642611.642630