PalmTouch: Using the Palm as an Additional Input Modality on Commodity Smartphones

Huy Viet Le[1], Thomas Kosch[2], Patrick Bader[1], Sven Mayer[1], Niels Henze[1]
[1]VIS, University of Stuttgart, Stuttgart, Germany
[2]LMU Munich, Munich, Germany
[1]{huy.le, patrick.bader, sven.mayer, niels.henze}@vis.uni-stuttgart.de, [2]thomas.kosch@ifi.lmu.de

ABSTRACT
Touchscreens are the most successful input method for smartphones. Despite their flexibility, touch input is limited to the location of taps and gestures. We present *PalmTouch*, an additional input modality that differentiates between touches of fingers and the palm. Touching the display with the palm can be a natural gesture since moving the thumb towards the device's top edge implicitly places the palm on the touchscreen. We present different use cases for *PalmTouch*, including the use as a shortcut and for improving reachability. To evaluate these use cases, we have developed a model that differentiates between finger and palm touch with an accuracy of 99.53 % in realistic scenarios. Results of the evaluation show that participants perceive the input modality as intuitive and natural to perform. Moreover, they appreciate *PalmTouch* as an easy and fast solution to address the reachability issue during one-handed smartphone interaction compared to thumb stretching or grip changes.

ACM Classification Keywords
H.5.2 User Interfaces: Input devices and strategies

Author Keywords
Palm; capacitive image; machine learning; smartphone.

INTRODUCTION
Smartphones have recently become the most successful mobile devices. Through a touchscreen, smartphones offer a wide range of functions that are used by millions of people. While most functions are accessible within a number of touches, some are used so frequently that shortcuts were introduced. Traditional devices offer volume buttons and a power button to enable users to change the device's volume and state with just one press. With an increasing number of frequently used functions such as device assistants, cameras or music players, device manufacturers and developers look for new ways to integrate them for faster access. Recently, Samsung introduced a dedicated hardware button to call the Bixby assistant on the Samsung Galaxy S8. The HTC U11 incorporates Edge Sense, a pressure sensitive frame that launches a user-defined action when squeezing the device. While these additional input controls fulfill their purpose, they require additional hardware which leaves out the possibility to update already existing and older devices. Further, these solutions clutter the device itself and become inconvenient when users are not holding the device in their hand or are using a bumper case.

(a) (b) (c)

Figure 1. Using the palm as an additional input modality on smartphones. Figure (a) shows a palm touch when holding the device one-handed, Figure (b) and (c) show palm touches for two-handed interaction.

Touch gestures constitute a possible solution to the challenge described above. For example, OnePlus devices recognize gestures on the lock screen to launch user-defined applications. However, system-wide drawn gestures that are always accessible for user-defined actions may conflict with other applications. Previous work presented a number of alternative input modalities to support traditional multi-touch input. This includes using the finger's 3D orientation [37, 41, 42, 50], contact size [7], pressure [23], or the shear force [21]. While these enrich the information of a finger's touch, they also bring restrictions since specific finger postures may now trigger unwanted actions. One solution to lower the likelihood of triggering unwanted actions is to differentiate between fingers or parts of fingers, which prevents interference with the main finger for interaction. Previous work [13, 16, 22] differentiated between different parts of the finger (*e.g.*, knuckle) or fingers themselves to assign unique touch actions.

Motivated by previous work, we applied this concept for one-handed as well as two-handed smartphone interaction. As a result, we present *PalmTouch*, an additional input modality that enables people to use the palm to trigger pre-defined functions instead of simply rejecting palm input as recent smartphones do. We show that this is a natural and fast gesture especially

CHI 2018, April 21–26, 2018, Montreal, QC, Canada

ISBN 978-1-4503-5620-6/18/04…$15.00

DOI: https://doi.org/10.1145/3173574.3173934

when the device is held one-handed. Stretching the thumb towards the top edge to access targets that are out of reach often places the palm on the touchscreen implicitly and subtly as shown in Figure 1a. The placement is often caused by unawareness of users which suggests that this gesture can be performed naturally. Figure 1 shows three examples of using *PalmTouch* in one-handed and two-handed scenarios to trigger assigned functions.

Previous work presented different features to detect a palm on a touchscreen. These include spatiotemporal touch features [43], and hand model filters [47] to detect the palm in inking scenarios on tablets. Moreover, Matero and Colley [36] presented characteristic patterns of unintentional touches, including touch duration which had the largest influence on rejection performance. However, these approaches require at least two touch points (pen and palm) or introduce latency due to temporal features which makes them not suitable for our proposed palm input modality. Recent smartphones feature a basic palm rejection which omits input in case the contact area is larger than a usual finger. However, they work on a driver level and are not reliable enough to be used for interaction.

In this work, we propose *PalmTouch*, an additional touch input modality to trigger pre-defined functions by placing the palm on the touchscreen. Accordingly, we present four use cases for *PalmTouch* and evaluate the input modality as a shortcut and to improve reachability during one-handed smartphone interaction. To evaluate *PalmTouch*, we have developed a palm detection model that differentiates between finger touches and palm touches with a high accuracy. In contrast to previous work, we use the raw capacitive image of the touchscreen to classify the low-resolution fingerprint using a convolutional neural network. We show that this runs on off-the-shelf smartphones, also works with single touch points and introduces no latency opposed to previous work. The contribution of this work is two-fold: (1) *PalmTouch*, an additional input modality using the palm, and (2) a high-accuracy palm detection model including a validation in realistic scenarios.

RELATED WORK

Previous work presented palm rejection approaches to ignore all touches made by the palm. Aiming to extend the touch input vocabulary by the palm, we also looked into a body of work that extended the input vocabulary on touchscreens.

Extending Touch Vocabulary

Previous work proposed different methods to extend the input vocabulary on touchscreens. Using capacitive images of touchscreens as shown in Figure 4, previous work [37, 50] trained machine learning models to estimate the 3D orientation of fingers on the screen. Based on the same technology, Holz *et al.* [25] presented *BodyPrint* to recognize body parts (*e.g.*, ear or fist) for user identification purposes. Similarly, Guo *et al.* [19] proposed to use capacitive images as a low-resolution fingerprint scanner to authenticate users. Recently, researchers built smartwatch prototypes which provide capacitive images to investigate area-based touches [39] and specific fingers in exaggerated poses as an additional input modality [16]. Similarly, Boring *et al.* [8] used the contact radius provided by iPhones as an additional input modality. Before Apple introduced *Force Touch* in 2014 which extended touch by a pressure modality, researchers presented a wide range of use cases for pressure-based touch, including one-handed zooming [38], and enriching touchscreen interaction by using shear which is the force tangential to a screen's surface [21]. Commercial and proprietary products include EarSense by Qeexo[1] which recognizes an ear on the touchscreen as a software-only alternative to the standard proximity sensor. Further, Qeexo's FingerSense is featured in the Huawei P10 as a shortcut to the screenshot function using the finger's knuckle. Harrison *et al.* [22] identified hand parts that are touching the screen by evaluating tap sounds from an object's impact. Similarly, Colley and Häkkilä [13] explored the feasibility of finger specific interaction on smartphones using a Leap Motion.

Based on additional hardware sensors, Wilkinson *et al.* [48] used a wrist-worn Inertial Measurement Unit (IMU) to add additional features such as roll, pitch, and force to touchscreen interaction. Similarly, Yeo *et al.* [52] combined the built-in accelerometer with touch input to enable one-handed text entry on large devices. Previous work also attached additional sensors to smartphones, such as Back-of-Device (BoD) sensors and input mechanisms on the edge of the device [33]. BoD interaction extends the input capability of the front display and enables a wide range of use cases [32], including addressing the fat-finger problem [4], improving reachability [31], 3D object manipulation [2, 44], preventing shoulder surfing through authenticating on the rear [14], and performing user-defined gesture input [45]. Input mechanisms on the device's edge serve as a novel way to zoom and scroll using pressure [24], or to select items while using the device one-handed [49]. While the described input modalities extend the touch input vocabulary, none of them focused on using different parts of the hand or finger for interaction using only off-the-shelf smartphones with a low likelihood of unintended activation.

Palm Rejection

Previous work presented algorithms to reject palm touches while writing on tablets using a pen. These algorithms are based on 2D touch points provided by the operating system. Schwarz *et al.* [43] identified different features to train a machine learning model for differentiating palm from pen touches. Features include touch duration, segmentation of touch points, consistency in touched areas, and the movement. Similarly, Tanyag *et al.* [47] used hand model filters to infer touches made by the palm. As these approaches require multiple touch points at once (palm and pen), they are limited to a writing scenario on tablets. For smartphones, Matero and Colley [36] identified a set of features to reject unintended touches. Amongst others, this includes the distance from the display edge and the duration which would introduce latency when used for classification. Recent Android smartphones feature a basic palm rejection which can be observed when generating a contact area on the touchscreen which is larger than a usual finger. However, this cannot be used for *PalmTouch* since it works on a driver level and does not work when the palm on the screen center as tested on different devices.

[1] www.qeexo.com - last access 2018-01-02

On the hardware side, smartphone manufacturers use inductive pens to differentiate stylus input from the capacitive input of the human finger. For example, Samsung's *S Pen*[2] uses an electromagnetic field generated from a compatible device (*e.g.*, Samsung Galaxy Note 2) to calculate the position relative to the screen. Liang *et al.* [35] proposed *GaussSense* which uses a grid of magnetic sensors on the back of the device to sense a magnetic field emitted from the stylus on the touchscreen. Gu *et al.* [18] used proximity sensing to recognize unintended palm touches on laptop computers. Recently, Camilleri *et al.* [11] investigated the effect of palm rejection technology on users. Their findings conclude with a less perceived discomfort and an increased productivity due to fewer interruptions.

Previous work invested considerable effort into palm detection approaches. However, the results were used to reject palm input instead of using it as an intended and additional input modality. Moreover, researchers used capacitive images on off-the-shelf devices to differentiate between different input objects. However, these were used either for authentication or on smartwatches. Using capacitive images to distinguish between finger and palms, we close the gap between these fields by proposing palm input as an additional input modality for one-handed and two-handed smartphone interaction.

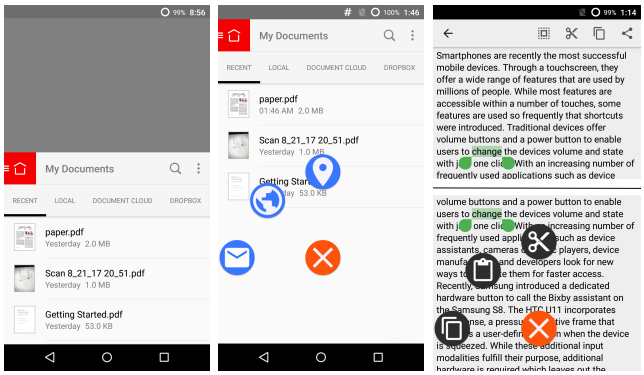
PALMTOUCH CONCEPT AND USE CASES

PalmTouch is an additional input modality for a wide range of functions. We applied the idea of hand part specific touch interaction presented in previous work (*e.g.*, using different fingers [13, 16] or finger parts [22]) for one-handed as well as two-handed interaction scenarios. Since using other fingers than the thumb or other parts of the hand (such as a knuckle) can be inconvenient or even infeasible during one-handed interaction, we instead use the palm for interaction.

During one-handed interaction, the palm can be placed subtly on the touchscreen by moving the thumb towards the upper edge of the device while stabilizing the device with fingers on the left edge as shown in Figure 1a. Since we use the palm of the same hand that is holding the smartphone, we refer to this movement as a *same-side palm touch*. During two-handed interaction, *PalmTouch* can be used by placing the flat hand (see Figure 1b) or by forming a fist on the touchscreen (see Figure 1c). Since we use the opposite hand to the one holding the device, we refer to this movement as an *opposite-side palm touch* based on the terminology used by Kerber *et al.* [29]. In the following, we present four use cases and discuss further input dimensions that extend the *PalmTouch* input modality.

Improving Reachability during One-Handed Interaction

Large smartphones pose challenges in reachability since they require changing the hand grip when used one-handed. With *PalmTouch*, users can stretch the thumb towards the top as if they would tap the target. This action implicitly places the palm on the touchscreen and can be used by *PalmTouch* to shift down the screen by half its size. A screen shift is exemplarily shown in Figure 2a and is similar to the iPhone's *Reachability* feature that can be activated by a double tap on the home button. Similarly, instead of dragging down the notification bar



(a) (b) (c)

Figure 2. Use cases for *PalmTouch*. Figure (a) demonstrates how *PalmTouch* improves reachability by moving down the screen by half its size; Figure (b) shows the pie menu for application launching and Figure (c) shows the pie menu for clipboard management.

which poses the same reachability challenge on large smartphones, *PalmTouch* can be used to open the notification drawer. Further difficult to reach UI elements include toolbars (*e.g.*, ActionBar[3]), URL bars in most browsers, search bars, menu buttons, and tabs.

Custom Actions and Applications

Smartphone manufacturers recently integrated simple and binary input modalities such as an extra button (Bixby button on the Samsung Galaxy S8) or a squeeze on the device's edge (Edge Sense on the HTC U11) to launch pre-defined applications. While these features require additional hardware, *PalmTouch* can be readily deployed onto recent and older off-the-shelf smartphones, *e.g.*, through software updates. Moreover, with the emergence of edge-to-edge displays on devices such as the iPhone X and Samsung Galaxy S8, the lack of a home button can be compensated with *PalmTouch*.

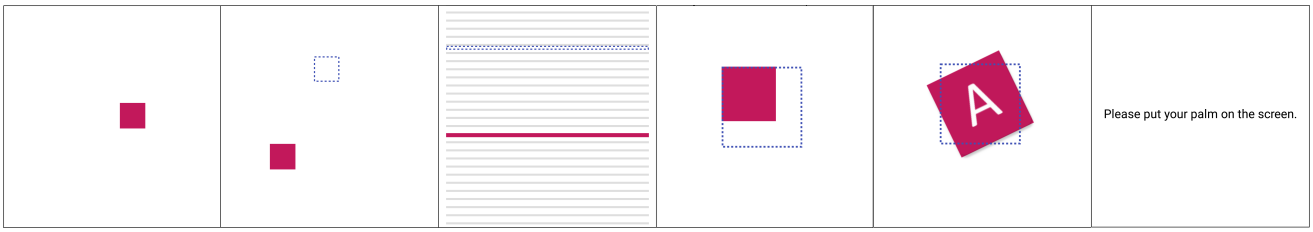
Instead of launching a single pre-defined action or application, a pie menu as shown in Figure 2b can be used to provide multiple options. The arrangement of buttons in a pie menu further benefits one-handed interaction. Previous work [6, 34] showed that the range of the thumb on a touchscreen is parabolic around the *carpometacarpal joint* (CMC) of the thumb. The CMC is located in the lower part of the palm. Since the palm is placed on the touchscreen to launch the pie menu, the thumb is already in a suitable position to tap the menu items. *PalmTouch* can also be used for application-dependent functions. For example, a palm touch could send away a message in a messaging application, while it accepts a call in the phone application or switch layers in Maps or CAD application. Since *PalmTouch* can be used eyes-free similar to a hardware button or squeeze, actions such turning off the screen or accepting a call can be mapped to a palm touch.

Clipboard Management

Copying and pasting from the clipboard are common actions in text editing tasks. While computer keyboards provide simple shortcuts, touch-based operating systems such as Android

[2] www.samsung.com/global/galaxy/galaxy-note5/spen

[3] developer.android.com/design/patterns/actionbar.html



(a) Tapping task (b) Dragging task (c) Scrolling task (d) Pinching task (e) Rotating task (f) Palm task

Figure 3. All six tasks performed by participants in the data collection study.

and iOS handle the access through context menus or buttons in the toolbar. A context menu requires a long press that takes between 500 *ms* and 1000 *ms* and could further move the caret to another location unintentionally due to the fat-finger problem [4]. Toolbar buttons require additional screen space. Therefore, we propose *PalmTouch* as a shortcut to the clipboard menu which avoids long-pressing and altering the caret position. To paste text, PalmTouch can open a menu which offers the function without a long-press. For text selection and copy/cut, users can perform a palm touch to start text selection and then use the menu as soon as the selection via finger was done to avoid a long-press. Figure 2c shows an example where users can select between copy, paste and cut after placing the palm on the touchscreen.

Unlocking the Device

PalmTouch can be used to unlock the smartphone by placing the palm on the touchscreen. This action can be done with a same-side palm touch while holding the device, or with one of the opposite-side variants when the device is, *e.g.*, lying on a table. In addition to the palm detection, *PalmTouch* can be extended to use the biometric features presented in *BodyPrint* [25] for authentication based on the capacitive images.

Additional Input Dimensions

In addition to a binary action, *PalmTouch* offers further dimensions that can be used for interaction. The contact area's centroid can be used as a proxy for the palm touch location. This enables the implementation of directional gestures, such as swiping up with the opposite hand's palm to exit an app and swiping left or right to switch between apps. The location of the opposite hand's palm can also be used to map functions to different locations of the touchscreen. For example, a palm touching the top half of the display skips to the next music title while a touch on the lower half plays the previous title. The location can also be used for a same-side palm touch (*e.g.*, x-position describes the used hand) to launch different actions depending on the hand that performed the palm touch.

DATA COLLECTION STUDY

To implement the use cases presented in the previous section, the touchscreen needs to differentiate between finger and palm touches. Previous work used the 2D touch location provided by the touchscreen which is either limited through latency [36, 43] or requires at least two touch points (pen and palm) [43, 47]. In contrast, we use capacitive images provided by the touchscreen which contain low-resolution fingerprints of the touch (*e.g.*, finger or palm). Since we apply machine learning to classify the touch, we conducted a user study to collect labeled touch data of fingers and the palm while participants perform representative touch actions. With this data, we train and evaluate a palm detection model to differentiate between touches from fingers and palms.

Study Design & Tasks

The purpose of this study is to collect a wide variety of finger and palm touch input. We designed six different tasks which instruct each participant to perform a total number of 240 representative touch actions. The first five tasks in Figure 3 (*finger tasks*) require participants to use their fingers whereas the rightmost task (*palm task*) instructs participants to place their palm on the screen (see Figure 1). The order of the finger tasks was randomized, and a palm task was performed after each *finger task* in an alternating order. Each *finger task* was performed 15 times.

Participants performed these tasks in two conditions, ONE-HANDED with the thumb as the main input finger and TWO-HANDED with the index finger as the main input finger. We conducted these tasks to capture different finger and palm touches in our data set. In both conditions, participants had to perform *tapping*, *dragging*, and *scrolling* movements. The TWO-HANDED condition also cover *zooming* and *pinching* movements. After each task, participants placed their palm on the touchscreen until they were told to remove it by the apparatus. We counterbalanced the variant of the opposite-side palm touch between participants (*flat hand* and *forming a fist*). We instructed participants to place their palm as if that would activate an function, such as launching the application drawer.

Participants & Study Procedure

We recruited 22 participants (5 female) between the ages of 21 and 34 ($M = 25.1$, $SD = 3.2$). All participants were right-handed. The average hand size was measured from the wrist crease to the middle fingertip, and ranged from 17.0 *cm* to 21.9 *cm* ($M = 19.2\,cm$, $SD = 1.6\,cm$). Our collected data comprise samples from the 5th and 95th percentile of the anthropometric data reported in prior work [40]. Thus, the sample can be considered as representative.

After participants signed the consent form, we measured their hand size. We then explained the procedure including the palm input modality as shown in Figure 1 and handed them an instruction sheet which explains all tasks of the study. We asked participants to repeatedly try out the movements until

they felt comfortable to repeat a palm touch at any given moment. Participants performed all tasks in 20 minutes on average and were rewarded with sweets for their participation.

Apparatus

We used an LG Nexus 5 running Android 5.1.1 with a modified kernel to access the 15×27 raw capacitive image of the Synaptics ClearPad 3350 touch sensor (see Figure 4). Each image pixel corresponds to a $4.1\,mm \times 4.1\,mm$ square on the $4.95''$ touchscreen. The pixel values represent the differences in electrical capacitance (in pF) between the baseline measurement and the current measurement. We developed an application for the tasks described above which logs a capacitive image every $50\,ms$ ($20\,fps$). Each image is logged with the respective task name so that every touch is automatically labeled.

PALMTOUCH MODEL AND IMPLEMENTATION

Based on the collected data set, we train a model to classify touches as being made by a finger or a palm. We will first show simple approaches based on feature engineering and established machine learning algorithms known from previous HCI work. Afterwards, we show that representation learning techniques such as Neural Networks (NNs) and Convolutional Neural Networks (CNNs) outperform the simpler approaches regarding the classification accuracy. Models and test results are summarized in Table 1.

Data Set & Preprocessing

After filtering empty (due to not touching) and erroneous images (which do not contain the expected number of touches) to avoid wrong labeling, we have a data set comprising $138,223$ capacitive images which represent blobs of valid touches. We extended the data set with flipped versions (vertical, horizontal, and both) of all remaining capacitive images to train the model for different device orientations. To train a position-invariant model and enable classification of multiple blobs within one capacitive image, we performed a blob detection, cropped the results and pasted each blob into an empty 15×27 matrix (referred to as *blob image*). The blob detection omitted all blobs that were not larger than one pixel of the image ($4.1\,mm \times 4.1\,mm$) as these can be considered as noise

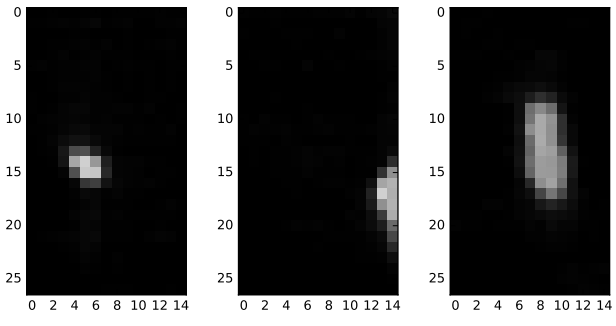


Figure 4. Exemplary raw capacitive images from the data collection study. Figure (a) shows the finger of participant 5 during the dragging task; (b) shows the palm of participant 19 in the one-handed condition and (c) shows the palm of participant 9 in the two-handed condition.

of the capacitive touchscreen. In total, our data set consists of $552,892$ blob images. We trained and tested all models with a participant-wise split of 80% to 20% (18:4) to avoid samples of the same participant being in both training and test set.

Overview of Basic Machine Learning Approaches

Due to their prominence in previous HCI work (*e.g.*, [22, 46, 51]), we trained and evaluated palm touch models based on Support Vector Machines (SVMs), k-nearest neighbors (kNNs), Decision Trees (DTs) and Random Forests (RFs). In contrast to representation learning approaches [5], these algorithms require the training data to be processed into features (*i.e.*, feature engineering). Using scikit-learn 0.18.2[4], we trained different models and performed a grid search as proposed by Hsu *et al.* [26] to determine the most suitable hyperparameters. If we did not report a hyperparameter, we applied the standard value as reported in scikit-learn's documentation. Since the palm's average contact area on the touchscreen ($M = 932.06\,mm^2$, $SD = 503.17\,mm^2$) is larger than the finger's ($M = 164.86\,mm^2$, $SD = 50.77\,mm^2$), we first used the blob area as a single feature to classify the touch. We determined the blob area by fitting an ellipse around the blob[5]. With an accuracy of 96.80% (prec = 97.66%; rec = 92.05%), the DT (with $max\_depth = 4$ to avoid overfitting) achieved the highest accuracy of the aforementioned algorithms. After experimenting with a wide range of additional features including the ellipse parameters and the capacitance represented by the blob, we found that a feature set comprising the ellipse (area, width and height) and the capacitance (mean and sum) achieved the highest accuracy of 98.17% (prec = 96.10%, rec = 98.18%) using an RF.

Representation learning algorithms learn features in part with the labeled input data and have been shown to be more successful than manual feature engineering for image data [5]. Thus, we implemented a multilayer feedforward neural network using *TensorFlow*[6] and performed a grid search over different network configurations, including the number of neurons in steps of 50, layers in steps of 1, activation functions, and optimizers provided by *TensorFlow*. Our final network architecture is shown in Table 1. Training was done with a batch size of 100 using the Adaptive Gradient Algorithm (AdaGrad) [15] with an adaptive learning rate starting from 0.001. We initialized the network weights using the Xavier initialization scheme [17]. While we experimented with L2 Regularization and batch normalization [27], this did not improve the overall accuracy. We achieved an accuracy of 98.74% (prec = 97.49%, rec = 98.46%) with this network configuration.

PalmTouch using a Convolutional Neural Network

CNNs are the recent state-of-the-art method for image classification [30]. As blobs are represented by low-resolution images, we implemented a CNNs using *TensorFlow*. We performed a grid search over the number of layers, filters and their sizes in steps of 1, the number of neurons in the fully

[4] scikit-learn.org/0.18/documentation.html

[5] 2D least squares estimator for ellipses: scikit-image.org/docs/dev/api/skimage.measure.html#skimage.measure.EllipseModel

[6] www.tensorflow.org/

connected layer in steps of 50, as well as activation functions and optimizers provided by *TensorFlow*. Our final network architecture is shown in Table 1. We trained the CNN using AdaGrad as the optimizer with a batch size of 100 and used the Xavier initialization scheme to initialize the network weights. We initialized the biases with random values from a normal distribution. An exponential decay (rate = 0.2 in 1000 steps) was used to decrease the initial learning rate of 0.009. We used L2 Regularization to compensate overfitting by adding 0.01 of the weights to the cost function. Moreover, we used an early stopping approach as proposed by Caruana *et al.* [12] to further avoid overfitting. While we experimented with batch normalization [27], this did not improve the overall accuracy. Our CNN achieved an accuracy of 99.58% (prec = 99.38%, rec = 99.28%) which is the highest of all presented approaches.

Mobile Implementation

After freezing the CNN to a protocol buffer file, we used *TensorFlow Mobile*[7] for Android to run the CNN on an LG Nexus 5 that provides the same capacitive images as in the data collection study. Classifying one capacitive image including the blob detection takes 7.5 *ms* on average (*min* = 4 *ms*, *max* = 11 *ms*, *SD* = 1.6 *ms*) over 1000 runs. As this is faster than the sampling rate for the capacitive images, it can be used to classify each sample when running in the background. With processor manufacturers recently starting to optimize their processors for machine learning (*e.g.*, Snapdragon 835), the classification can be sped up significantly[8]. The model can be further optimized for mobile devices with techniques such as quantization [20] and pruning [1] for a small loss of accuracy.

Discussion

We presented an overview of machine learning algorithms which we used to train a palm classifier and showed that a CNN achieved the highest classification accuracy of 99.58%. This improves the baseline accuracy by 31.0%. While our grid search already yields reasonable results for the basic machine learning approaches, further optimizing accuracy and especially precision is necessary as the palm classifier is supposed to run in the background to classify a large number of input frames over time (*i.e.*, 20 frames per second). As fingers are used most of the time to perform input on the touchscreen while a detected palm triggers a defined action, false positives (affecting the precision score) lead to a visible unexpected behavior. In contrast, a lower recall score (and thus a higher false negative rate) could be partly compensated by the UI through, *e.g.*, recovering previous wrongly classified palm touches as soon as the palm is correctly detected. Thus, we prioritized precision over recall in the training process. While the SVM with ellipse and capacitance properties as features achieved the highest precision of all approaches, the trade-off is by far the lowest recall and also accuracy. In total, the CNN achieved the best results while the preparation and classification are feasible to perform on an off-the-shelf mobile device. We will refer to this model as *CNN-PalmTouch*.

[7] www.tensorflow.org/mobile/

[8] www.qualcomm.com/news/onq/2017/01/09/tensorflow-machine-learning-now-optimized-snapdragon-835-and-hexagon-682-dsp

features	algorithm	parameters / layers	prec	rec	acc
–	Baseline (ZeroR)	Always predicting `Finger` as this is the majority class.	-	0.0	68.54
finger blob size	kNN	*k* (*neighbors*) = 197	97.90	91.78	96.80
	DT	*max depth* = 4	97.66	92.05	96.80
	RF	*estimators* = 1; *max depth* = 1	97.64	92.06	96.80
	SVM	*C* = .1; linear kernel	98.99	90.54	96.73
ellipse & capacitance	kNN	*k* (*neighbors*) = 9	98.40	94.34	97.73
	DT	*max depth* = 6	97.10	96.70	98.05
	RF	*estimators* = 16; *max depth* = 10	96.10	98.18	98.17
	SVM	*C* = 10; linear kernel	99.96	79.06	93.40
raw data (RL)	NN	*Input*: 405 *Hidden Layer 1*: 500 (ReLU) *Hidden Layer 2*: 300 (ReLU) *Softmax (output)*: 2	97.49	98.46	98.74
	CNN	*Input*: $27 \times 15 \times 1$ *Convolution*: $7 \times 7 \times 16$ (ReLU) *Max Pooling*: 2×2 (stride = 2) *Convolution*: $7 \times 7 \times 36$ (ReLU) *Max Pooling*: 2×2 (stride = 2) *FC Layer 1*: 350 (ReLU) *FC Layer 2*: 350 (ReLU) *Softmax (output)*: 2	99.38	99.28	99.58

Table 1. Performance of the trained models with the hyperparameters after a grid search for the highest accuracy. Results (in percent) were calculated using the test set described above.

EVALUATION OF PALMTOUCH

We conducted a study to evaluate *PalmTouch* and the model accuracy in realistic scenarios. Specifically, we focus on the following three aspects: 1) classification accuracy of *CNN-PalmTouch* in realistic scenarios, 2) qualitative feedback after using *PalmTouch*, and 3) a quantitative evaluation of the reachability use case. We used a Nexus 5 running the mobile version of *CNN-PalmTouch* described above and custom applications to record and implement the study scenarios.

Study Procedure & Design

We designed four tasks to evaluate the three aspects described above. We measured the participants' hand and finger sizes after we obtained informed consent and then handed them an instruction sheet that explained all parts of the study so that participants could refer to the instructions at any time.

Part 1 (Realistic Scenario for Evaluating False Positives)

In a counterbalanced order, we instructed participants to perform tasks one-handed and two-handed which we refer to as realistic scenarios. While participants used the smartphones, we collected the classifier output in the background to test the model on false positives as palms are not expected in this part. We designed the realistic scenarios to cover commonly used touch input gestures including tapping, dragging, scrolling, and additionally pinching and rotating for two-handed use. To keep the scenarios as realistic as possible, participants performed this part on a pure Android system using common applications such as the onboard SMS messenger, Google Chrome, and Maps.

We handed the Nexus 5 in standby mode to the participant who received a (simulated) notification after unlocking the device. Tapping the message in the notification drawer then opens a text message with questions that the participant needs to answer by using Google searches or Maps. We further provided an instruction sheet that describes each step that the

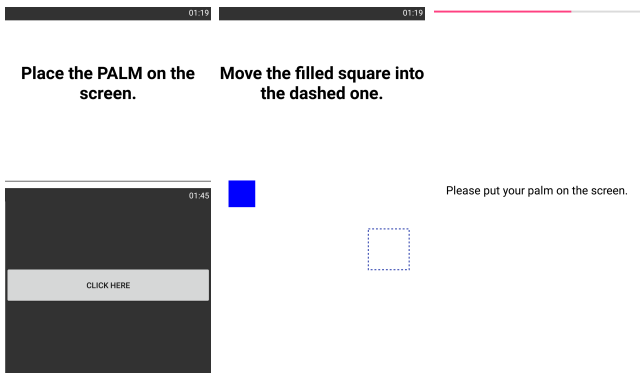
participant is required to do. The gestures described above were used especially using the system (tapping and scrolling), selecting text on a website (long press and dragging), and navigating in Google Maps in the two-handed scenario (pinching and rotating). Each of the two scenarios ended with replying to the initial SMS message with the search results. In total, this part took 10 minutes per participant.

Part 2 (Realistic Scenario for Qualitative Feedback)
We introduced and demonstrated *PalmTouch* to the participants and let them practice the same-side and one of the opposite-side palm touches using a demo application. Afterwards, participants performed a modified version of the Part 1 scenarios. Instead of pulling down the notification bar, participants now use the palm to access the notifications. Further, we replaced all application switching actions with the pie menu containing the messaging, browser and maps application. After completion, participants filled out a questionnaire and we interviewed them about their impression of *PalmTouch*. In total, this part took around 15 minutes per participant.

Part 3 (Reachability Task)
We evaluated the reachability use case regarding the task completion time (TCT) and qualitative feedback. Specifically, we compared accessing notifications supported by *PalmTouch* with dragging down the notification bar manually. We used a 2×2 within-subjects design with the independent variables being the number of hands (ONE-HANDED and TWO-HANDED) and the access method (PALM and DRAG). Each condition comprised 20 repetitions of opening the notification drawer to click on a notification displayed at a random height. Between these repetitions, participants completed 2 - 5 Fitts' Law tasks as shown in Figure 5b to ensure that they returned to their usual hand grip after clicking the notification. We measured the TCT for opening the notification drawer and clicking on the notification. We further collected qualitative feedback about the perceived easiness, speed, success, accuracy and comfort using a 7-point Likert scale.

The apparatus simulates a notification bar (see Figure 5a) for the respective conditions. To simulate the DRAG condition as realistic as possible, the notification drawer can also be opened with a fling downwards. By rooting the Nexus 5, we disabled Android's notification bar and navigation bar to avoid any disruptions during this part. This part took 10 minutes.

Part 4 (Palm Touch Input for Evaluating False Negatives)
We tested the model on false negatives. Our study application instructed participants to repeatedly place their palm on the screen for one second and remove it afterward (see Figure 5c). The duration ensures that participants contribute a similar number of samples and avoids palm touches being done too quickly or slowly. Both same-side and opposite-side palm touches were performed 20 times each in a counterbalanced order. We collected the classifier output during this part to test the model on false negatives as fingers are not expected in this part. We let participants perform this part at the end of the study since repeatedly placing the palm on the touchscreen and waiting could lead to fatigue and therefore influence the other parts. This part took 5 minutes.



(a) (b) (c)

Figure 5. Screenshots of (a) the notification drawer in Part 3; (b) Fitts' Law task as a distraction task in Part 3; and (c) a prompt to place the palm on the touchscreen until the progressbar on top is full (Part 4).

Participants

We recruited 22 participants (6 female) with an average age of 21.9 (SD = 2.1) who had not participated in the previous study. All except two participants were right-handed. The average hand size measured from the wrist crease to the middle fingertip ranged from 17.2 cm to 20.5 cm (M = 18.6 cm, SD = 0.9 cm). Three participants preferred to use their smartphone two-handed, while 14 preferred to use it one-handed and five use both variants in everyday life.

RESULTS

We present the results of the evaluation study which covers model accuracy, evaluation of the reachability use case, heatmaps of how *PalmTouch* was used, and qualitative feedback.

Model Accuracy in Realistic Scenarios

We obtained labeled capacitive images of touch input in a realistic scenario. With the labels providing us with the number of false positives and true negatives (Task 1), and true positives and false negatives (Task 4), we derived the precision, recall, and accuracy of the classifiers. After calculating all three metrics for each participant, the mean accuracy yielded by *CNN-PalmTouch* is 99.53 % (SD = 0.71 %). The mean precision is 99.35 % (SD = 2.53 %) and the mean recall is 97.94 % (SD = 2.87 %). The false positive rate which describes the likelihood of unintentionally triggering a palm input is 0.09 %.

Reachability Use Case Evaluation

For the ONE-HANDED condition, the average time for DRAG to open the notification drawer is 1689.62 ms (SD = 700.32 ms) while the total time including tapping the notification is 2352.74 ms on average (SD = 817.14 ms). In contrast, the average time for PALM to open the notification drawer is 1396.31 ms (SD = 449.75 ms) and 2114.73 ms (SD = 647.14 ms) including tapping the notification. A two-tailed paired t-Test revealed a significant difference in TCT to open the notification drawer in the PALM and DRAG condition (t_{329} = 6.71, $p < .001$) and in TCT including tapping the notification (t_{329} = 4.40, $p < .001$). We used a Wilcoxon signed-rank test to analyze the Likert scores as shown in Table 2 for the ONE-HANDED condition and found a statistically

significant difference between PALM and DRAG in perceived easiness ($Z = -2.201$, $p = .028$), speed ($Z = -1.985$, $p = .047$), success ($Z = -2.069$, $p = .039$) and accuracy ($Z = -2.087$, $p = .037$). No statistically significant difference was found for the perceived comfort ($Z = -.508$, $p = .612$).

For the TWO-HANDED condition, the average time for DRAG to open the notification drawer is 1462.54 *ms* (*SD* = 873.14 *ms*) while the total time including tapping the notification is 2103.94 *ms* on average (*SD* = 1049.05 *ms*). In contrast, the average time for PALM to open the notification drawer is 1394.29 *ms* (*SD* = 588.76 *ms*) and 2110.40 *ms* (*SD* = 742.37 *ms*) including tapping the notification. A two-tailed paired t-Test showed neither a significant difference in TCT between the PALM and DRAG condition to open the notification drawer ($t_{329} = 1.19$, $p = .236$) nor in the TCT including tapping the notification ($t_{329} = -0.09$, $p = .925$). We used a Wilcoxon signed-rank test to analyze the Likert scores for the TWO-HANDED condition and did not find a statistically significant difference between PALM and DRAG neither in perceived easiness ($Z = -.626$, $p = .531$), speed ($Z = -.019$, $p = .985$), success ($Z = -1.562$, $p = .118$), accuracy ($Z = -.894$, $p = .371$), nor comfort ($Z = -1.326$, $p = .185$).

Type and Location of Palm Placement

Figure 6 shows heatmaps of the locations at which participants performed palm touches in task 4, and indicate the touches' average position. All three images represent the average capacitive images over each participant. We separated the capacitive images into three palm input types that we showed in Figure 1: same-side (as shown in Figure 1a), opposite-side using the flat hand (Figure 1b), and opposite-side by forming a fist (Figure 1c). Nine participants decided to use the flat hand for opposite-side palm touch and 13 participants the fist.

Qualitative Feedback in Realistic Scenarios

Participants filled out a SUS questionnaire [9] about *PalmTouch* as an additional input modality in the two realistic scenarios. SUS scores range between 0 to 100 whereas any score above 68 is considered to be above average in terms of usability [10]. The SUS score for *PalmTouch* ranged from 52.5 to 95.5 with an average of 80.1 (*SD* = 10.0). With this, the

(a) (b) (c)

Figure 6. Average capacitive images representing the location of the palm touches for (a) same-side, (b) opposite-side using the flat hand, and (c) opposite-side by forming a fist. All images describe the average capacitive images over each participant.

	One-Handed		Two-Handed	
Perception	**Palm**	**Drag**	**Palm**	**Drag**
Easiness *	5.7 (1.1)	4.8 (1.6)	6.0 (1.2)	5.6 (1.7)
Speed *	5.6 (1.2)	4.6 (1.8)	5.4 (1.6)	5.2 (1.8)
Success *	6.0 (1.4)	5.1 (1.6)	6.2 (1.0)	5.5 (1.6)
Accuracy *	5.7 (0.9)	5.0 (1.4)	5.9 (1.0)	5.4 (1.6)
Comfort	4.5 (1.6)	4.5 (1.9)	6.1 (1.0)	5.4 (1.5)

Table 2. Subjective perceptions (7-point Likert scale) to open the notification drawer in both conditions. Values in brackets indicate the SD, an asterisk (*) indicate a statistically significant difference between one-handed PALM and DRAG ($p < .05$).

average score lies between "good" and "excellent" in the adjective rating of Bangor *et al.* [3]. Realistic scenarios without *PalmTouch* yielded an SUS score of *M* = 74.0 (*SD* = 16.1).

When asked about the first impression after using *PalmTouch* in realistic scenarios, the majority (18) were positive about using the palm as an additional input modality. Especially when used one-handed, participants found using the palm intuitive and natural (P7, P10, P11, P12, P13, P14, P20), comfortable (P3, P6, P9) and convenient (P2, P3, P13). While participants needed a short period of time to get familiar with the input modality, they (P12, P13, P15, P21) appreciate that it helps them to use the system faster than before ("*It felt strange for a short while, but then I became familiar with it really fast. After that, it feels intuitive, and I am faster than without it.* " - P21). Moreover, they were surprised about the stability of the system ("*I was surprised that the system worked really well, especially for app switching.*" - P13; "*It worked well*" - P9). In contrast, four participants reportedly had concerns to perform a palm touch ("*I was afraid to drop the phone*" - P22) or "*had the feeling that [they] touch something on the screen unintentionally*" when used the first time (P17).

When asked about advantages of *PalmTouch*, eight participants reportedly find the provided shortcuts useful. They identified that these shortcuts provide faster access to apps (P9, P11, P12, P17, P18, P19) and improve reachability, especially when using the device one-handed (P7, P10, P20). As an example for the faster access, P16 explained that the most important apps are "*always available when placing the palm onto the touchscreen*". Further, P7 suggested that "*systems could use palm input to [allow users to] access the app drawer from every screen*" to make launching apps faster. Three participants (P7, P10, P20) argued that *PalmTouch* saves grip changes as "*shortcuts help to reach buttons on the top side of the touchscreen*" (P20).

We asked participants about perceived disadvantages of *PalmTouch*. For *PalmTouch* in the one-handed condition, only two participants (P3, P7) reported that when "*tapping something on the upper left edge of the device, one could accidentally place the palm on the screen*" (P3) which could be solved with the reachability use case. In general, participants see more advantages than disadvantages for *PalmTouch* when they use the device one-handed. In contrast, they reported more disadvantages after the two-handed scenarios. Holding the device with one hand and placing the palm of the second hand on the touchscreen feels unintuitive (P2, P12, P16) and unnatural (P5, P6, P7, P11). As an example, P12 explained that "*switching*

from index finger to the palm requires either moving the hand or turning it" which makes it inconvenient. Further, three participants (P3, P20, P22) argued that it is faster to use the index finger of the second hand to reach targets on the top side of the touchscreen instead of the palm. Since two-handed interaction does not pose reachability challenges, participants found that *PalmTouch* was less useful in the two-handed scenarios.

We asked participants for further scenarios in which *PalmTouch* can be useful. They preferred and suggested the possibility to start custom applications and actions (P2, P3, P6, P11, P14, P15, P17, P18), such as the camera (P2, P6), settings within an application (P11, P17) or splitting the screen (P18) which is shipped with Android 7.0. P1 and P22 even suggested mapping more critical functions since they find it unlikely to trigger a function mapped to the palm accidentally. These functions include closing the foreground application (P1), accepting a call (P20), or stopping the music (P22).

DISCUSSION

We implemented *PalmTouch* and deployed it on an off-the-shelf smartphone to enable users to trigger specific functions by placing the palm on the touchscreen. The palm as an additional input modality received a SUS score of 80.1 which is considered above average in terms of usability [10]. The SUS score conforms with subjective feedback of participants who found *PalmTouch* intuitive and natural as a way to improve reachability and as a shortcut. Using the notification bar as an abstract scenario of the reachability problem, we found that participants perceive *PalmTouch* as significantly easier, faster and more accurate than changing the grip which could lead to dropping the device. For the one-handed scenarios, an analysis of the task completion time (TCT) revealed that *PalmTouch* is indeed significantly faster than a grip change to open the notification drawer manually. This finding can be transferred to other interface elements such as toolbars, the application's menu button, and URL bars in most browsers. Further, with the emergence of edge-to-edge displays on devices such as the iPhone X and Samsung Galaxy S8, the lack of a dedicated home button can be compensated with *PalmTouch*.

Participants gave more positive feedback for *PalmTouch* during the one-handed scenario. The reason is that two-handed interaction does not pose any reachability challenges since the interacting hand can move freely over the whole display. Moreover, placing the other hand's palm on the display feels reportedly less subtle and thus can feel unusual. In both scenarios, all except two participants had no difficulties to place their palms on the touchscreen after a short practice phase. Due to small hand sizes (17 *cm*), two participants lack a stable grip while holding the device one-handed. Moreover, bending the *thenar muscles*[9] to place the palm on the touchscreen causes the hand to bend. Thus, all other fingers move towards the palm which leads to an unstable grip while the device is tilted. In this situation, a controllable input is not possible since the device needs to be balanced. However, this also applies to stretching the thumb or changing the grip. Thus, we recommend *PalmTouch* as an additional input modality while still providing alternative touch input in case the user cannot ensure a stable hand grip.

[9]Thenar muscles refers to a group of muscles located at the base of the thumb [28].

We implemented *PalmTouch* using capacitive images of the touchscreen and trained a CNN which achieves a high accuracy in detecting the palm. Compared to basic machine learning approaches and neural networks, the CNN achieved the highest accuracy with an applicable classification time when deployed on an off-the-shelf LG Nexus 5. We showed that our model classifies touches reliably during realistic scenarios with an accuracy of 99.53%. With a precision of 99.35%, the likelihood of unintended triggers either through classification errors or unwanted palm input is neglectable. With a recall of 97.94%, our classifier also recognized the palm reliably when users placed them on the screen. False negatives were caused by capacitive images in which the palm was about to be placed on the screen (in Task 4), and can be corrected by recovering previous wrongly classified palm touches as soon as the palm is correctly detected. The accuracy can be further improved by taking the average locations of palm input as shown in Figure 6 into account. Since accuracies in offline validation and realistic scenarios are similar, this shows that our model is generalizing well and does not overfit. In summary, this shows that using the palm to activate functions is feasible with a high accuracy while perceived as natural and fast by users.

LIMITATIONS

While we observed that all touches which are larger than a usual finger are rejected on the touchscreen of the LG Nexus 5, we did not modify its palm rejection algorithm to use the results of our classifier. Thus, there were palm touches that are correctly classified by our model but were not rejected by the touch controller. However, this effect was not disruptive in our realistic scenarios as non-rejected touches only slightly scrolled the shown website or moved the map.

While we showed a neglectable likelihood of unintended triggers, the existence of low-level palm rejection algorithms might suggest that unintended touches occur more frequently, *e.g.*, in in-the-wild scenarios. Palm rejection algorithms are designed to omit unintended touches whereas our *PalmTouch* model is trained to detect intended palm touches as performed in the data collection study. When the device is used normally, activations through unintended palm touches are unlikely as we showed in the evaluation. However, it is left to future work to further investigate this aspect in scenarios where all kind of different touches might happen, such as while walking, being encumbered, and being in crowded places.

CONCLUSION

We presented *PalmTouch*, an additional input modality on smartphones using the palm to perform input. We proposed four use cases and evaluated *PalmTouch* in a user study. Participants perceived *PalmTouch* as a natural and intuitive gesture for a wide range of use cases, including the use as a shortcut and to improve reachability in one-handed scenarios. We investigated an abstract scenario in which we addressed reachability issues during one-handed smartphone interaction and found that *PalmTouch* was perceived as an easier, faster and more accurate solution than a grip change which could

drop the device. While a quantitative analysis revealed that participants were indeed faster with *PalmTouch*, they appreciated its short learning curve. Especially on recent devices with an edge-to-edge display (*e.g.*, iPhone X), *PalmTouch* provides an alternative to the removed home button.

We implemented *PalmTouch* using capacitive images collected in a controlled study, and a convolutional neural network to differentiate between touches being made by fingers and palms. In contrast to previous work, our approach uses low-resolution fingerprints instead of heuristics that only work with multiple touch points (*i.e.*, pen and palm) or that would introduce latency through temporal features. This enables us to build a model with an accuracy of 99.53 % in a realistic scenario evaluation. Since we only modified the software of an off-the-shelf Nexus 5 smartphone, *PalmTouch* could be readily deployed onto recent smartphones, *e.g.*, through software updates.

We focused on the feasibility and usability of *PalmTouch* as an input modality on smartphones. With a precision of 99.35 %, we showed a neglectable likelihood of unintended triggers either through classification errors or unintended palm touches. Future work could evaluate *PalmTouch* in the wild and investigate unintended palm touches. The palm detection model could be extended to differentiate between unintended and intended palm touches to support the device's native palm rejection and to improve the accuracy of intended palm touches.

PALMTOUCH DATASET AND MODEL

One outcome of the studies is a labeled dataset that consists of capacitive images representing touches from fingers and palms. We are publicly releasing the data set together with Python 3.6 scripts to preprocess the data as well as train and test the model as described in this paper. We further provide the trained model as a protocol buffer file, the software to run *PalmTouch*, and implementations of the use cases readily deployable on Android. These enable the community to run *PalmTouch* on their devices: http://github.com/interactionlab/PalmTouch.

ACKNOWLEDGEMENTS

This work is supported through project C04 of SFB/Transregio 161, the MWK Baden-Württemberg within the Juniorprofessuren-Programm, and by the DFG within the SimTech Cluster of Excellence (EXC 310/2).

REFERENCES

1. Sajid Anwar, Kyuyeon Hwang, and Wonyong Sung. 2017. Structured Pruning of Deep Convolutional Neural Networks. *J. Emerg. Technol. Comput. Syst.* 13, 3, Article 32 (Feb. 2017), 18 pages. DOI: http://dx.doi.org/10.1145/3005348
2. Patrick Bader, Valentin Schwind, Niels Henze, Stefan Schneegass, Nora Broy, and Albrecht Schmidt. 2014. Design and Evaluation of a Layered Handheld 3D Display with Touch-sensitive Front and Back. In *Proceedings of the 8th Nordic Conference on Human-Computer Interaction: Fun, Fast, Foundational (NordiCHI '14)*. ACM, New York, NY, USA, 315–318. DOI:http://dx.doi.org/10.1145/2639189.2639257
3. Aaron Bangor, Philip Kortum, and James Miller. 2009. Determining What Individual SUS Scores Mean: Adding an Adjective Rating Scale. *J. Usability Studies* 4, 3 (May 2009), 114–123. http://dl.acm.org/citation.cfm?id=2835587.2835589
4. Patrick Baudisch and Gerry Chu. 2009. Back-of-device Interaction Allows Creating Very Small Touch Devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '09)*. ACM, New York, NY, USA, 1923–1932. DOI: http://dx.doi.org/10.1145/1518701.1518995
5. Yoshua Bengio, Aaron Courville, and Pascal Vincent. 2013. Representation Learning: A Review and New Perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35, 8 (2013), 1798–1828. DOI: http://dx.doi.org/10.1109/TPAMI.2013.50
6. Joanna Bergstrom-Lehtovirta and Antti Oulasvirta. 2014. Modeling the Functional Area of the Thumb on Mobile Touchscreen Surfaces. In *Proceedings of the 32Nd Annual ACM Conference on Human Factors in Computing Systems (CHI '14)*. ACM, New York, NY, USA, 1991–2000. DOI: http://dx.doi.org/10.1145/2556288.2557354
7. Sebastian Boring, David Ledo, Xiang 'Anthony' Chen, Nicolai Marquardt, Anthony Tang, and Saul Greenberg. 2012a. The Fat Thumb: Using the Thumb's Contact Size for Single-handed Mobile Interaction. In *Proceedings of the 14th International Conference on Human-computer Interaction with Mobile Devices and Services (MobileHCI '12)*. ACM, New York, NY, USA, 39–48. DOI:http://dx.doi.org/10.1145/2371574.2371582
8. Sebastian Boring, David Ledo, Xiang 'Anthony' Chen, Nicolai Marquardt, Anthony Tang, and Saul Greenberg. 2012b. The Fat Thumb: Using the Thumb's Contact Size for Single-handed Mobile Interaction. In *Proceedings of the 14th International Conference on Human-computer Interaction with Mobile Devices and Services (MobileHCI '12)*. ACM, New York, NY, USA, 39–48. DOI:http://dx.doi.org/10.1145/2371574.2371582
9. John Brooke. 1996. SUS-A quick and dirty usability scale. *Usability evaluation in industry* 189, 194 (1996), 4–7.
10. John Brooke. 2013. SUS: A Retrospective. *J. Usability Studies* 8, 2 (Feb. 2013), 29–40. http://dl.acm.org/citation.cfm?id=2817912.2817913
11. Matt J Camilleri, Ajith Malige, Jeffrey Fujimoto, and David M Rempel. 2013. Touch displays: the effects of palm rejection technology on productivity, comfort, biomechanics and positioning. *Ergonomics* 56, 12 (2013), 1850–1862. DOI: http://dx.doi.org/10.1080/00140139.2013.847211 PMID: 24134774.
12. Rich Caruana, Steve Lawrence, and C Lee Giles. 2001. Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. In *Advances in neural information processing systems*. 402–408.

13. Ashley Colley and Jonna Häkkilä. 2014. Exploring Finger Specific Touch Screen Interaction for Mobile Phone User Interfaces. In *Proceedings of the 26th Australian Computer-Human Interaction Conference on Designing Futures: The Future of Design (OzCHI '14)*. ACM, New York, NY, USA, 539–548. DOI: http://dx.doi.org/10.1145/2686612.2686699

14. Alexander De Luca, Emanuel von Zezschwitz, Ngo Dieu Huong Nguyen, Max-Emanuel Maurer, Elisa Rubegni, Marcello Paolo Scipioni, and Marc Langheinrich. 2013. Back-of-device Authentication on Smartphones. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*. ACM, New York, NY, USA, 2389–2398. DOI: http://dx.doi.org/10.1145/2470654.2481330

15. John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *Journal of Machine Learning Research* 12 (July 2011), 2121–2159. http://dl.acm.org/citation.cfm?id=1953048.2021068

16. Hyunjae Gil, DoYoung Lee, Seunggyu Im, and Ian Oakley. 2017. TriTap: Identifying Finger Touches on Smartwatches. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17)*. ACM, New York, NY, USA, 3879–3890. DOI: http://dx.doi.org/10.1145/3025453.3025561

17. Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *In Proceedings of the International Conference on Artificial Intelligence and Statistics. Society for Artificial Intelligence and Statistics (AISTATS'10)*, Vol. 9. JMLR.org, 249–256. http://www.jmlr.org/proceedings/papers/v9/glorot10a.html

18. Jiseong Gu, Seongkook Heo, Jaehyun Han, Sunjun Kim, and Geehyuk Lee. 2013. LongPad: A Touchpad Using the Entire Area Below the Keyboard of a Laptop Computer. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*. ACM, New York, NY, USA, 1421–1430. DOI: http://dx.doi.org/10.1145/2470654.2466188

19. Anhong Guo, Robert Xiao, and Chris Harrison. 2015. CapAuth: Identifying and Differentiating User Handprints on Commodity Capacitive Touchscreens. In *Proceedings of the 2015 International Conference on Interactive Tabletops & Surfaces (ITS '15)*. ACM, New York, NY, USA, 59–62. DOI: http://dx.doi.org/10.1145/2817721.2817722

20. Song Han, Huizi Mao, and William J. Dally. 2015. Deep Compression: Compressing Deep Neural Network with Pruning, Trained Quantization and Huffman Coding. *CoRR* abs/1510.00149 (2015). http://arxiv.org/abs/1510.00149

21. Chris Harrison and Scott Hudson. 2012. Using Shear As a Supplemental Two-dimensional Input Channel for Rich Touchscreen Interaction. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '12)*. ACM, New York, NY, USA, 3149–3152. DOI: http://dx.doi.org/10.1145/2207676.2208730

22. 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. DOI: http://dx.doi.org/10.1145/2047196.2047279

23. Seongkook Heo and Geehyuk Lee. 2011. Forcetap: Extending the Input Vocabulary of Mobile Touch Screens by Adding Tap Gestures. In *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services (MobileHCI '11)*. ACM, New York, NY, USA, 113–122. DOI:http://dx.doi.org/10.1145/2037373.2037393

24. David Holman, Andreas Hollatz, Amartya Banerjee, and Roel Vertegaal. 2013. Unifone: Designing for Auxiliary Finger Input in One-handed Mobile Interactions. In *Proceedings of the 7th International Conference on Tangible, Embedded and Embodied Interaction (TEI '13)*. ACM, New York, NY, USA, 177–184. DOI: http://dx.doi.org/10.1145/2460625.2460653

25. 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. DOI: http://dx.doi.org/10.1145/2702123.2702518

26. Chih-Wei Hsu, Chih-Chung Chang, Chih-Jen Lin, and others. 2003. A practical guide to support vector classification. (2003).

27. Sergey Ioffe and Christian Szegedy. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *CoRR* abs/1502.03167 (2015). http://arxiv.org/abs/1502.03167

28. L.A. Jones and S.J. Lederman. 2006. *Human Hand Function*. Oxford University Press. https://books.google.de/books?id=InyYROA6j_0C

29. Frederic Kerber, Pascal Lessel, and Antonio Krüger. 2015. Same-side Hand Interactions with Arm-placed Devices Using EMG. In *Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems (CHI EA '15)*. ACM, New York, NY, USA, 1367–1372. DOI: http://dx.doi.org/10.1145/2702613.2732895

30. Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*. 1097–1105.

31. Huy Viet Le, Patrick Bader, Thomas Kosch, and Niels Henze. 2016. Investigating Screen Shifting Techniques to Improve One-Handed Smartphone Usage. In *Proceedings of the 9th Nordic Conference on Human-Computer*

Interaction (NordiCHI '16). ACM, New York, NY, USA, Article 27, 10 pages. DOI: http://dx.doi.org/10.1145/2971485.2971562

32. Huy Viet Le, Sven Mayer, Patrick Bader, Frank Bastian, and Niels Henze. 2017. Interaction Methods and Use Cases for a Full-Touch Sensing Smartphone. In *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems (CHI EA '17)*. ACM, New York, NY, USA, 2730–2737. DOI: http://dx.doi.org/10.1145/3027063.3053196

33. 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. DOI: http://dx.doi.org/10.1145/3098279.3122143

34. Huy Viet Le, Sven Mayer, Patrick Bader, and Niels Henze. 2018. Fingers' Range and Comfortable Area for One-Handed Smartphone Interaction Beyond the Touchscreen. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI'18)*. ACM, New York, NY, USA. DOI: http://dx.doi.org/10.1145/3173574.3173605

35. Rong-Hao Liang, Kai-Yin Cheng, Chao-Huai Su, Chien-Ting Weng, Bing-Yu Chen, and De-Nian Yang. 2012. GaussSense: Attachable Stylus Sensing Using Magnetic Sensor Grid. In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology (UIST '12)*. ACM, New York, NY, USA, 319–326. DOI: http://dx.doi.org/10.1145/2380116.2380157

36. Juha Matero and Ashley Colley. 2012. Identifying Unintentional Touches on Handheld Touch Screen Devices. In *Proceedings of the Designing Interactive Systems Conference (DIS '12)*. ACM, New York, NY, USA, 506–509. DOI: http://dx.doi.org/10.1145/2317956.2318031

37. 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. DOI: http://dx.doi.org/10.1145/3132272.3134130

38. Takashi Miyaki and Jun Rekimoto. 2009. GraspZoom: Zooming and Scrolling Control Model for Single-handed Mobile Interaction. In *Proceedings of the 11th International Conference on Human-Computer Interaction with Mobile Devices and Services (MobileHCI '09)*. ACM, New York, NY, USA, Article 11, 4 pages. DOI: http://dx.doi.org/10.1145/1613858.1613872

39. Ian Oakley, Carina Lindahl, Khanh Le, DoYoung Lee, and MD. Rasel Islam. 2016. The Flat Finger: Exploring Area Touches on Smartwatches. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*. ACM, New York, NY, USA, 4238–4249. DOI: http://dx.doi.org/10.1145/2858036.2858179

40. A Poston. 2000. Human engineering design data digest. *Washington, DC: Department of Defense Human Factors Engineering Technical Advisory Group* (2000).

41. Simon Rogers, John Williamson, Craig Stewart, and Roderick Murray-Smith. 2011. AnglePose: Robust, Precise Capacitive Touch Tracking via 3D Orientation Estimation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '11)*. ACM, New York, NY, USA, 2575–2584. DOI: http://dx.doi.org/10.1145/1978942.1979318

42. Anne Roudaut, Eric Lecolinet, and Yves Guiard. 2009. MicroRolls: Expanding Touch-screen Input Vocabulary by Distinguishing Rolls vs. Slides of the Thumb. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '09)*. ACM, New York, NY, USA, 927–936. DOI: http://dx.doi.org/10.1145/1518701.1518843

43. Julia Schwarz, Robert Xiao, Jennifer Mankoff, Scott E Hudson, and Chris Harrison. 2014. Probabilistic palm rejection using spatiotemporal touch features and iterative classification. In *Proceedings of the 32Nd Annual ACM Conference on Human Factors in Computing Systems (CHI '14)*. ACM, New York, NY, USA, 2009–2012. DOI: http://dx.doi.org/10.1145/2556288.2557056

44. Erh-li Early Shen, Sung-sheng Daniel Tsai, Hao-hua Chu, Yung-jen Jane Hsu, and Chi-wen Euro Chen. 2009. Double-side Multi-touch Input for Mobile Devices. In *CHI '09 Extended Abstracts on Human Factors in Computing Systems (CHI EA '09)*. ACM, New York, NY, USA, 4339–4344. DOI: http://dx.doi.org/10.1145/1520340.1520663

45. Shaikh Shawon Arefin Shimon, Sarah Morrison-Smith, Noah John, Ghazal Fahimi, and Jaime Ruiz. 2015. Exploring User-Defined Back-Of-Device Gestures for Mobile Devices. In *Proceedings of the 17th International Conference on Human-Computer Interaction with Mobile Devices and Services (MobileHCI '15)*. ACM, New York, NY, USA, 227–232. DOI: http://dx.doi.org/10.1145/2785830.2785890

46. Jie Song, Gábor Sörös, Fabrizio Pece, Sean Ryan Fanello, Shahram Izadi, Cem Keskin, and Otmar Hilliges. 2014. In-air Gestures Around Unmodified Mobile Devices. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology (UIST '14)*. ACM, New York, NY, USA, 319–329. DOI: http://dx.doi.org/10.1145/2642918.2647373

47. Riyeth P Tanyag and Rowel O Atienza. 2015. Implicit Palm Rejection Using Real-Time Hand Model Filters on Tablet Devices. In *9th International Conference on Next Generation Mobile Applications, Services and Technologies (NGMAST'15)*. IEEE, 347–352. DOI: http://dx.doi.org/10.1109/NGMAST.2015.45

48. Gerard Wilkinson, Ahmed Kharrufa, Jonathan Hook, Bradley Pursglove, Gavin Wood, Hendrik Haeuser, Nils Y. Hammerla, Steve Hodges, and Patrick Olivier. 2016. Expressy: Using a Wrist-worn Inertial Measurement Unit to Add Expressiveness to Touch-based Interactions. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*. ACM, New York, NY, USA, 2832–2844. DOI: http://dx.doi.org/10.1145/2858036.2858223

49. Graham Wilson, David Hannah, Stephen Brewster, and Martin Halvey. 2012. Investigating One-handed Multi-digit Pressure Input for Mobile Devices. In *CHI '12 Extended Abstracts on Human Factors in Computing Systems (CHI EA '12)*. ACM, New York, NY, USA, 1727–1732. DOI: http://dx.doi.org/10.1145/2212776.2223700

50. 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. DOI: http://dx.doi.org/10.1145/2817721.2817737

51. Hui-Shyong Yeo, Gergely Flamich, Patrick Schrempf, David Harris-Birtill, and Aaron Quigley. 2016. RadarCat: Radar Categorization for Input & Interaction. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology (UIST '16)*. ACM, New York, NY, USA, 833–841. DOI: http://dx.doi.org/10.1145/2984511.2984515

52. Hui-Shyong Yeo, Xiao-Shen Phang, Steven J. Castellucci, Per Ola Kristensson, and Aaron Quigley. 2017. Investigating Tilt-based Gesture Keyboard Entry for Single-Handed Text Entry on Large Devices. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17)*. ACM, New York, NY, USA, 4194–4202. DOI: http://dx.doi.org/10.1145/3025453.3025520