Force Touch Detection on Capacitive Sensors using Deep Neural Networks

Tobias Boceck
University of Stuttgart
Stuttgart, Germany
boceckts@gmail.com

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

Sascha Sprott
University of Stuttgart
Stuttgart, Germany
sprottsa@gmail.com

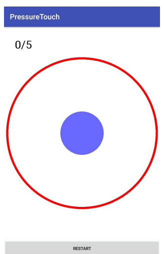
Sven Mayer [1,2]
[1]University of Stuttgart, Germany
[2]Carnegie Mellon University, US
info@sven-mayer.com

21st International Conference on Human-Computer Interaction with Mobile Devices and Services (MobileHCI '19), October 1–4, 2019, Taipei, Taiwan
ACM 978-1-4503-6825-4/19/10.
http://dx.doi.org/10.1145/3338286.3344389

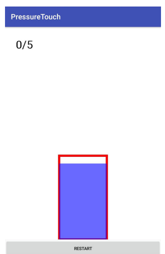
Abstract
As the touchscreen is the most successful input method of current mobile devices, the importance to transmit more information per touch is raising. A wide range of approaches has been presented to enhance the richness of a single touch. With Apple's *3D Touch*, they successfully introduce pressure as a new input dimension into consumer devices. However, they are using a new sensing layer, which increases production cost and hardware complexity. Moreover, users have to upgrade their phones to use the new feature. In contrast, with this work, we introduce a strategy to acquire the pressure measurements from the mutual capacitive sensor, which is used in the majority of today's touch devices. We present a data collection study in which we collect capacitive images where participants apply different pressure levels. We then train a Deep Neural Network (DNN) to estimate the pressure allowing for force touch detection. As a result, we present a model which enables estimating the pressure with a mean error of $369.0g$.

Author Keywords
Force touch; pressure; interaction; input dimension mutual; capacitive sensor; deep neural networks.

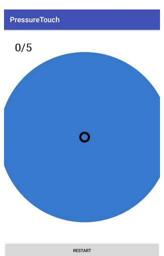
ACM Classification Keywords
H.5.2 [User Interfaces]: Haptic I/O



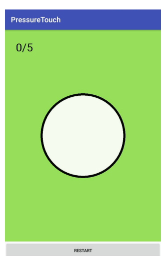
(a) Match circle



(b) Match bar



(c) Fill circle



(d) Fit color

Figure 1: For all for tasks the size or color is mapped to the pressure applied by the participants. The goal is to match them by varying the pressure.

Introduction & Background

With the iPhone 6s in 2015, Apple introduced *3D Touch*[1] which adds a pressure dimension to the input space. Asher Trockman ran a large-scale study to understand the capabilities of the sensor used by Apple[2]. In his report, he stated that the sensor's range is from $0g$ to $337g$, while the accuracy is not feasible to determine in such a large scale setting as ground-truth needs to be verified first. While this sensor, only covers a small range, the sensor needs to be built into the system upon production time. In Android systems, the MotionEvent is fitted with a pressure value already for years. Additional, the ForceTouch library[3] enables developers to add the pressure dimension to their prototypes easily. However, both rely on a pressure approximation based on the contact area of the finger, c.f. Boring et al. [2]. Thus, currently, true pressure input is not possible on Android systems. Arif and Stuerzlinger [1] present a method which does not rely on the simulated pressure, but on the fact that the contact area groups and therefore the touch point moves as the touch-controller predicts a different touch point over time when more pressure is applied.

In contrast, Ramos et al. [14] enabled pressure input using a pressure sensitive pen and found that the new dimension can enrich the input. Moreover, for stylus input, they found that dividing pressure space into six levels is optimal. Heo and Lee [5] presented an approach to use the accelerometer to detect pressure input but could only distinguish between two levels. Hwang et al. [7] improved the recognition and added a third level.

Inoue et al. [9] investigated using RGB images of a finger to estimate the applied pressure to the surface using DNNs. However, this approach makes pressure sensitive devices bulky and drastically reduces mobility. On the other hand, Philip Quinn [13] and Takada [16] used a barometric pressure sensors to estimate the force on touch screens.

More recently, pressure sensors have been deployed around the device, for example, the HTC U11 and the Google Pixel is designed with a feature called *Edge Sense*, which allows the user to press the frame of the device with its hand, to launch applications.

An alternative is to use the capacitive images to gain more information about the object that is touching it has a long history. Guarneri et al. [4] classified single finger, double fingers, and palm input using these images. Later, Le et al. [10] improve these results using DNN. Mayer et al. [12] also used a DNN and the capacitive images to estimate the finger orientation.

In this paper, we present a new approach to acquire the pressure input on a touchscreen. We developed a machine learning (ML) model, which can estimate the pressure put on a touchscreen by using capacitive images. We, therefore, first conducted a user study to collect training data.

Data Collection Study

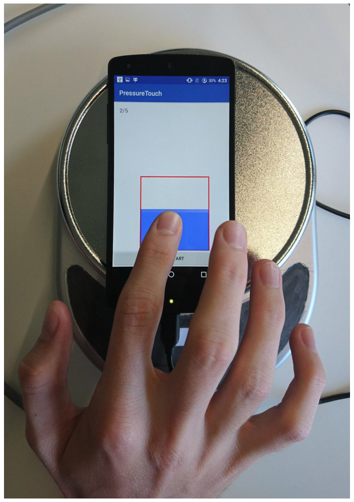
First, we collect labeled training data. As the goal is to estimate pressure a.k.a. the normal force[4] we need to record the applied pressure and the capacitive image. Thus, we run a study in which participants are asked to perform different pressure inputs.

[1] https://developer.apple.com/ios/3d-touch/

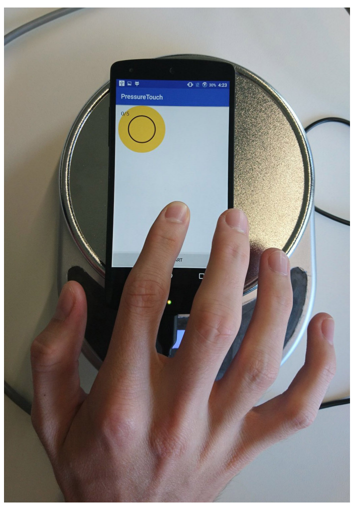
[2] https://ashertrockman.github.io/ios/2015/10/24/3d-touch-scale.html

[3] https://github.com/michelelacorte/ForceTouch

[4] The normal force is the force perpendicular to the surface. In our case the touchscreen.



(a) Match bar



(b) Match circle

Figure 2: Example images while a participant is performing the *match bar* and *fill circle* task.

Apparatus
We used three devices: an Android LG Nexus 5 which retrieves the capacitive images, a digital weighing scale to retrieve the pressure applied by the participants, and a laptop to sync the data using a local network with a delay of less than 1*ms*. The phone was only connected to the laptop minimizing shunting effects [8].

As typical consumer weighing scales cannot read the scale continuously at a high rate, we used the openscale[5] by SparkFun. After calibration, we were able to record the scale measurements every 84*ms*. 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 without background subtraction. The modified kernel was set up to capture a capacitive image every 50*ms*, c.f. Le et al. [11]. As a last step, we tared the scale after placing the phone on it.

Task
We implemented an app which was able to receive the weight measurements of the scale and record the weight with the corresponding capacitive image. We implemented four tasks (see Figure 1), each of which followed the same pattern, namely to match some shapes or colors by applying different pressure levels on the touchscreen. The design of the tasks was inspired by Hwang et al. [7]. In the *match circle* and *match bar* task (see Figures 1a and 1b), the goal was to match the red outline of a circle and bar with the blue corresponding circle and bar by applying different pressure levels. The *fill circle* task (see Figure 1c) was similar to the first ones *match circle* task; however, the position was varied and the pressure values were multiplied by a random factor smaller 1 to collect higher pressure samples.

[5]https://github.com/sparkfun/OpenScale

For the *Fit color* task (see Figure 1d), the color of the inner circle had to be matched to the color of the outer area. More pressure turned the color darker.

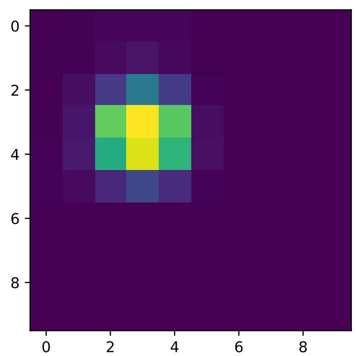
Procedure
The participants were instructed on the goal of the study and on how to use the data collection app. In the app, the participant was first asked to enter demographic data. Then the participants were told only to use their index finger to interact with the phone (see Figure 2). They complete 15 sets of randomly selected tasks. In each set participants were asked to complete the task five times.

Participants
We recruited participants from our university's volunteer pool. In total, 20 participants took part in the study (15 male, and 5 female). The age range was between 19 and 27 years ($M = 23.3$, *SD* $= 1.7$).

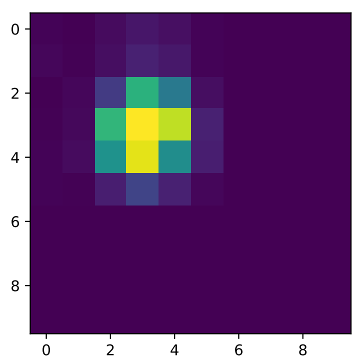
Machine learning (ML)

In our data collection study, we collected 115,134 samples where participants applied different pressure levels on the screen. We used mean squared error (MSE) as optimization function and report root mean squared error (RMSE) and mean absolute error (MAE) for readability in grams.

Pre-Processing
We first performed a blob detection using OpenCV to ensure a finger was present on the screen. We then removed all images with a time difference between capacitive image and weight measurement larger than 80ms and images with no pressure reading. This results in $82,526$ samples with an average blob size of 18.4*px*^{2} (*SD* $=$ 4.1*px*^{2}). As less than 1% of the data was with pressure values above $2,000g$, we removed all samples with more than $2,000g$. For the remaining data, we performed data augmentation to increase the data set size and equalizing the sample sizes



(a) 100g pressure



(b) 600g pressure

Figure 3: Two capacitive sample images of the capacitive matrix after pre-processing.

for low vs. large pressure values. We first added random Gaussian noise (*SD* $= 1.5$) to the images to boost the number of images per $50g$ bin to 7,000 images. This boosted the data set size to $280,000$ images. As final augmentation step, we flipped images horizontally and vertically to increase the sample size to $1,120,000$. Finally, we pasted the data in the upper left corner which makes our model position invariant (see Figure 3).

Baseline

First, we determined a baseline performance by using well-established ML models to determine the pressure. 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. [10]. Pearson's correlation revealed a significant correlation between the ellipse area and the pressure $p < .001$ with $\rho = 0.23$. Thus, we trained different basic ML models to set a baseline. Therefore, we performed a grid search with a 5-fold cross-validation (CV).

A *k*NN with k=6 performs best; however, it is the worst baseline with a *RMSE* $= 659.02g$ and *MAE* $= 534.56g$ (*SD* $= 385.45g$). DT (*maxDepth* $= 18$, *minSamplesSplit* $= 22$) *RMSE* $= 611.58g$, and *MAE* $= 492.70g$ (*SD* $= 362.33g$) was the next up. Runner up with *RMSE* $= 593.51g$, and *MAE* $= 511.86g$ (*SD* $= 298.72g$) is SVM. Finally the best baseline predictor is a RF with 14 estimators: *RMSE* $= 583.36g$, and *MAE* $= 470.51g$ (*SD* $= 344.88g$).

Convolutional Neural Network (CNN) Training

For training, we used a 75% to 25% (15:5) participant-wise split for train and test data set to avoid samples of the same participant being in both training and test set. We randomly picked 5 test participants, while with data augmentation they made up for 26% of the data, for testing, we only used the $16,781$ non augmented samples to reduce overfitting

	RMSE	**MAE**	**SD**
*k*NN	659.02	534.56	385.45
DT	611.58	492.70	362.33
SVM	593.51	511.86	298.72
RF	583.36	470.51	344.88
CNN	471.99	369.01	294.3

Table 1: Results of the pressure prediction. Here, we present the results of basic machine learning algorithms as well as the best CNN model of the test set.

towards the data argumentation. We implemented a DNN for regression using Keras 2.2.4 with TensorFlow 1.12 as backend. We applied the trial-and-error method [3] to find the best parameters for our models.

Representation learning algorithms learn features in part with the labeled input data and have been shown to be more successful than manual feature engineering. Thus, we implemented a multilayer feedforward DNN, which is shown in Figure 4. The training was done with a batch size of 500 using the RMSprop optimizer with a mechanism which reduces the learning rate by 10% when a metric has stopped improving over 10 epochs. We found that an initial learning rate of $.001$ leads to the best performance. We initialized the network weights using the Xavier initialization scheme. For the CNN layer, we set padding to be same, the kernel to be 3×3, and as an activation function, we used a ReLU function. We used dropout layers during training between all hidden layers with a dropout level of 50%. Further, we performed batch normalization after each CNN layer. Finally, for both fully connected dense layers we used LeakyReLU as activation function with a L1/L2 regularization of $.02$ and $.15$ respectively.

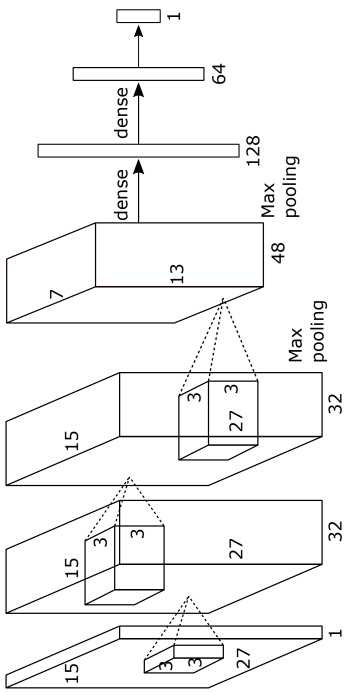


Figure 4: The best performing Convolutional Neural Network (CNN) structure to estimate the pressure put on a mutual capacitive sensor by a human finger.

Our CNN achieved an *MAE* $= 396.5g$ (*SD* $= 278.3g$) while the *RMSE* is $484.5g$. For the test set the model achieved *MAE* $= 369.0g$ (*SD* $= 294.3g$) with an *RMSE* of $472.0g$ after 500 epochs. Train results are expectantly worse due to the use of LeakReLU and Dropout, we did this to counteract overfitting when using our heavily augmented data set.

Discussion

While we improved over our baseline approaches, we did not achieve the same sensitivity as for instance, the dedicated pressure sensor in today's iPhones. However, Huber et al. [6] estimate a just-noticeable difference (JND) for in-car interaction at $118g$ (50%). Moreover, they state that two levels need to be at least $292g$ different to be considered different $(25\%$ rating), and $460g$ for 0%. Thus, our model is in the same range where humans can distinguish between different pressure levels.

We treated the problem as a regression problem to continuously estimate the pressure. However, with the current quality of our model, a continues pressure input is not feasible for interaction. We argue that we can use the current CNN to implement a classifier with two stats such as "normal press" and "force press" with pressure around $0g$ and $500g$ as target pressure level. Thus, when treating pressure as a classification problem with two stats we can enrich the interaction on mutual capacitive screens without adding a extra sensing layer. This can be used for single touches, and multi-touch, but also for force gesture input extending gestures sets by an extra dimension.

While we present a step towards an estimation of pressure without the need for additional hardware, there is still a remaining error in the estimation. One reason includes the limitation of the touch sensor of the LG Nexus 5. With a pixel size of 4.1×4.1*mm*, the capacitive image still has a low-resolution which restricts the performance of the estimation. This could improve when using precise high-resolution touch sensors. This technology is already available in commercial smartphones, c.f. Microsoft PixelSense.

Finally, we contribute the data set, Python scripts as well as a ready to deploy model at GitHub[6] under the MIT license to enable future evaluation with new models and other estimation approaches.

Conclusion and Future Work

We presented an approach to detect force touch through a finger onto a mutual capacitive sensor. First, we collected a data set and used state-of-the-art DNN techniques to outperform our baseline estimation. We trained a DNN with 3 CNN layers followed by 2 fully connected dense layers.

While our research enables force touch detection without an additional sensing layer, future work is needed to enable accurate pressure input. Thus, as the next steps, we aim to collect a larger dataset for performance improvements. Related work such as Schweigert et al. [15] shows that executing the model on a phone is feasible, however deploring the pressure model to a phone is a logical next step.

Acknowledgement

This work was supported by the German Research Foundation (DFG) within Cluster of Excellence in Simulation Technology (EXC 310/2) at the University of Stuttgart.

REFERENCES

1. Arif, A. S. and Stuerzlinger, W. Pseudo-pressure Detection and Its Use in Predictive Text Entry on Touchscreens. In Proc. of OzCHI '13. DOI: http://dx.doi.org/10.1145/2541016.2541024

[6] https://github.com/interactionlab/ForceTouchDetection

2. Boring, S., Ledo, D., Chen, X. A., Marquardt, N., Tang, A., and Greenberg, S. The Fat Thumb: Using the Thumb's Contact Size for Single-handed Mobile Interaction. In Proc. of MobileHCI '12. DOI: http://dx.doi.org/10.1145/2371574.2371582
3. Coulibaly, P., Anctil, F., and Bobée, B. B. Daily reservoir inflow forecasting using artificial neural networks with stopped training approach. *Journal of Hydrology* (2000). DOI: http://dx.doi.org/10.1016/S0022-1694(00)00214-6
4. Guarneri, I., Capra, A., Castorina, A., Battiato, S., and Farinella, G. M. PCA based shape recognition for capacitive touch display. In Proc. of ICCE '13. DOI: http://dx.doi.org/10.1109/ICCE.2013.6487033
5. Heo, S. and Lee, G. Forcetap: Extending the Input Vocabulary of Mobile Touch Screens by Adding Tap Gestures. In Proc. of MobileHCI '11. DOI: http://dx.doi.org/10.1145/2037373.2037393
6. Huber, J., Sheik-Nainar, M., and Matic, N. Multi-Level Force Touch Discrimination on Central Information Displays in Cars. In Proc. of AutomotiveUI '17. DOI: http://dx.doi.org/10.1145/3131726.3132043
7. Hwang, S., Bianchi, A., and Wohn, K.-y. VibPress: Estimating Pressure Input Using Vibration Absorption on Mobile Devices. In Proc. of MobileHCI '13. DOI: http://dx.doi.org/10.1145/2493190.2493193
8. Ikematsu, K., Fukumoto, M., and Siio, I. Ohmic-Sticker: Force-to-Motion Type Input Device for Capacitive Touch Surface. In Proc. of CHI EA '19. DOI: http://dx.doi.org/10.1145/3290607.3312936
9. Inoue, Y., Makino, Y., and Shinoda, H. Estimation of the Pressing Force from Finger Image by Using Neural Network. In Proc. of EuroHaptics '18.
10. Le, H. V., Kosch, T., Bader, P., Mayer, S., and Henze, N. PalmTouch: Using the Palm As an Additional Input Modality on Commodity Smartphones. In Proc. of CHI '18. DOI: http://dx.doi.org/10.1145/3173574.3173934
11. Le, H. V., Mayer, S., Bader, P., and Henze, N. A Smartphone Prototype for Touch Interaction on the Whole Device Surface. In Proc. of MobileHCI '17. DOI: http://dx.doi.org/10.1145/3098279.3122143
12. Mayer, S., Le, H. V., and Henze, N. Estimating the Finger Orientation on Capacitive Touchscreens Using Convolutional Neural Networks. In Proc. of ISS '17. DOI:http://dx.doi.org/10.1145/3132272.3134130
13. Quinn, P. Estimating Touch Force with Barometric Pressure Sensors. In Proc. of CHI '19. DOI: http://dx.doi.org/10.1145/3290605.3300919
14. Ramos, G., Boulos, M., and Balakrishnan, R. Pressure Widgets. In Proc. of CHI '04. DOI: http://dx.doi.org/10.1145/985692.985754
15. Schweigert, R., Leusmann, J., Hagenmayer, S., Weiß, M., Le, H. V., Mayer, S., and Bulling, A. KnuckleTouch: Enabling Knuckle Gestures on Capacitive Touchscreens using Deep Learning. In Proc. of MuC '19. DOI: http://dx.doi.org/10.1145/3340764.3340767
16. Takada, R., Lin, W., Ando, T., Shizuki, B., and Takahashi, S. A Technique for Touch Force Sensing Using a Waterproof Device's Built-in Barometer. In Proc. of CHI EA '17. DOI: http://dx.doi.org/10.1145/3027063.3053130