MetaDigit: Towards A Practical Digits Input System with Few User Effort

Shicong Hong⋆, Zhihong Xiao⋆, Zishuo Guo, Yongpan Zou, Kaishun Wu
{hongshicong2019,xiaozhihong2019}@email.szu.edu.cn;{yongpan,wu}@szu.edu.cn
College of Computer Science and Software engineering, Shenzhen University

ABSTRACT

Nowadays, smart devices have become increasingly essential in humans' life. However, traditional input methods may not work effectively due to the tiny screen of the devices. Moreover, most learning-based gesture input schemes only perform well in terms of certain metrics such as accuracy, without considering other aspects like response time and user training overhead which are essential for real-world usage scenarios. Without taking the trade-off between different metrics into account, existing learning-based gesture input systems suffer from severe performance degradation in practice. In this paper, we investigate the trade-off between evaluation metrics of mobile interaction systems and then report our attempt towards a more practical digits input system based on our previous work. We propose an acoustic-based device-free digits input system named MetaDigit which achieves over 85% accuracy of real-time digits recognition with even zero-shot from new users.

CCS CONCEPTS

• **Human-centered computing** → Interaction techniques; Gestural input.

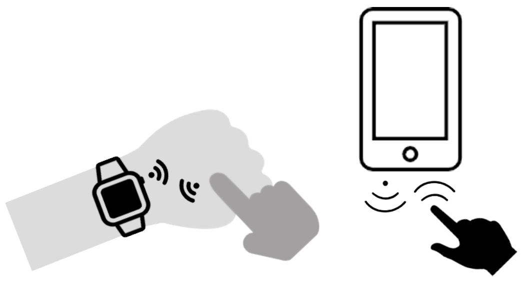
KEYWORDS

Smart devices; Gesture input; Acoustic sensing; Few-shot learning

ACM Reference Format:
Shicong Hong⋆, Zhihong Xiao⋆, Zishuo Guo, Yongpan Zou, Kaishun Wu. 2020. MetaDigit: Towards A Practical Digits Input System with Few User Effort. In *The 2nd International Workshop on Challenges in Artificial Intelligence and Machine Learning for Internet of Things (AIChallengeIoT'20), November 16–19, 2020, Virtual Event, Japan.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3417313.3429377

1 INTRODUCTION

With the rising popularity of tiny smart devices such as smartwatches and smart glasses, researchers have proposed plenty of novel gesture-based interaction methods to cope with the 'fat finger' problem on such devices. According to the signals utilized in these methods, they can mainly be categorized into speech recognition-based, radio-frequency (RF) signals-based [9], inertial sensors-based



(a) With Smartwatch. (b) With Smartphone.

Figure 1: The possible scenarios of AcousDigits.

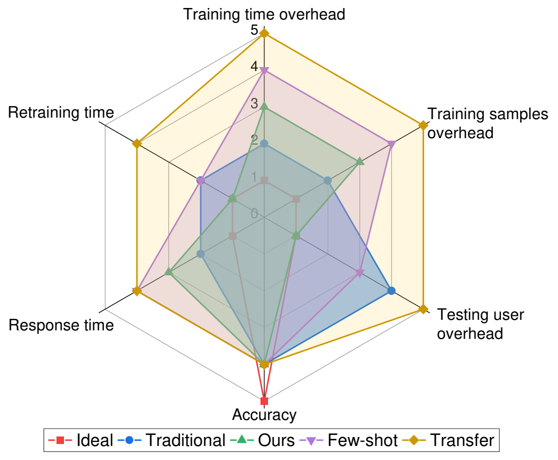


Figure 2: The evaluation metrics of a gesture-based HCI system

[3] and acoustic signal-based [11]. Among them, speech recognition possesses the risk of privacy leakage and social discomfort, especially in the public. RF-based methods require expensive and bulky signal transceivers, which makes them not appropriate for interaction on these devices. Inertial sensors-based schemes require users to hold the device in hand while performing actions [3]. In contrast, acoustic-based methods stand out by the merits of sensor pervasiveness, device-free working style and fine-grained resolution as shown in Fig. 1, due to which researchers have paid much attention to this emerging sensing technique.

Regardless of the hardware, most of the above systems share a similar learning-based data processing pipeline, either with machine learning algorithms or deep learning models, and report distinctive performance mainly in terms of accuracy. Nevertheless, we regard that there should be more considerations in evaluating

⋆ indicates that both authors contributed equally to this work.

AIChallengeIoT'20, November 16–19, 2020, Virtual Event, Japan
© 2020 Association for Computing Machinery.
ACM ISBN 978-1-4503-8134-5/20/11…$15.00
https://doi.org/10.1145/3417313.3429377

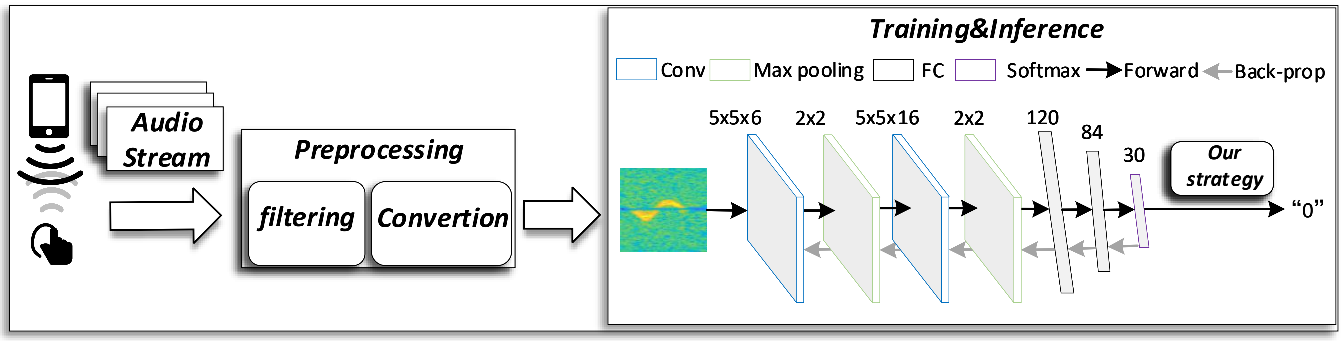


Figure 3: The data processing workflow of MetaDigit

a learning-based mobile interaction system except for recognition accuracy, especially considering practical usage cases. For example, the previous work [11] report a high accuracy of recognizing ten basic digits, but with the requirement of a new user providing about 15 samples for each digit to train the model. When the testing environment changes, new users need to collect much more data for model training, which obviously causes a heavy burden for users and degrades the user experience. This is the so-called cross-domain problem. Even though some latest works [5] proposed some latest deep learning [10] techniques such as transfer learning, few-shot learning and *etc.* to deal with this problem, they bring about overhead in other aspects such as collecting overwhelming training data, much longer training time, and slower real-time response, due to the complexity of latest techniques.

To gain more comprehensive knowledge about the evaluation of a learning-based mobile interaction system, we summarize the possible metrics needed to be considered as shown in Fig. 2. In the following, we give a brief introduction to these metrics.

- *Accuracy*: how accurate the system can recognize gestures?
- *User testing overhead*: how much effort a user needs to spare to use the system? This metric is quantified by the number of data samples that a new user needs to provide for training a system.
- *Response time*: how fast can the system give a response? This metric is quantified by the time that a system needs to give a response to a gesture.
- *Training samples overhead*: how much effort needs to be paid to collecting training data set? It is quantified by the number of samples that a system designer collects.
- *Training time overhead*: how much time does the model training process occupy? It is closely related to the complexity of a model.
- *Retraining time*: how long does it take to retrain the model? Retraining is a necessary process for transfer learning methods and some few-shot learning models.

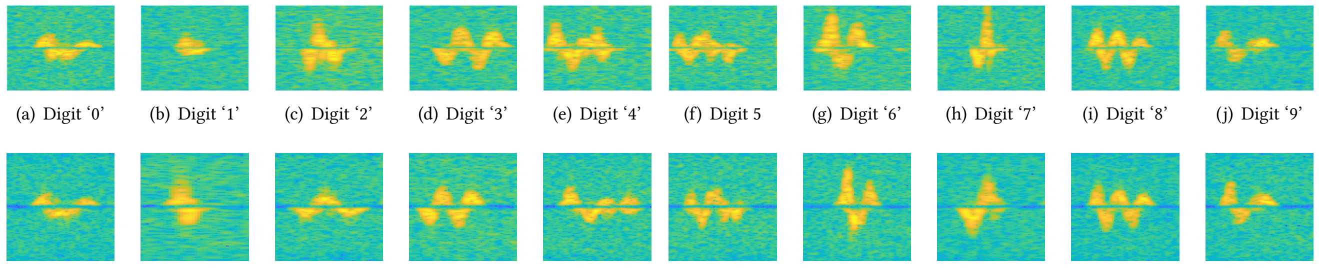
Ideally, a mobile interaction system should perform well in all the above metrics, that is, high accuracy, low user testing overhead, short latency and little training overhead. Unfortunately, it is usually rather difficult to achieve due to the trade-off between different metrics. For example, compared with traditional machine learning techniques, few-shot learning can be used to reduce user testing overhead, but at the cost of lower accuracy and longer training time. In a nutshell, existing works usually show favorable performance in certain aspects without considering the trade-off between different evaluation metrics, which makes them less practical in real-world usage cases.

The above observations motivate us to explore the question, *i.e.*, can we propose a digit input system, maintaining users device-free and requiring less training data or even no training data from new users? In this paper, we present a digits-entry system called MetaDigit which allows a user to input digits in the air via acoustic signals. Different from our earlier work [11], we focus on relaxing some constraints such as model retraining and user testing overhead to the system's practicability and user experience, making it more practical in real-world usage scenarios. As a first step, we investigate the shortcomings of existing methods in terms of the metrics listed in Fig. 2 and consider the trade-off between them. Then we apply some latest techniques such as few-shot learning, which have been reported to achieve high performance in recent works, to our task and analyze their performances. After that, we come up with our strategy which combines training data partitions with data augmentation, aiming to reduce training overhead of a new user but maintain comparable even better performance in other metrics compared to few-shot learning models. The remainder of this paper is organized as follows. Sec. 2 introduces the system design of MetaDigit. Sec. 3 demonstrates the experiments and evaluation. In Sec. 4, we discuss the future work towards designing a practical mobile interaction system. At last, we conclude this work in Sec. 5.

2 SYSTEM DESIGN

2.1 Sensing Signal

As mentioned above, in order to satisfy two requirements (*i.e.*, device-free and privacy protection), we make use of 19 kHz near ultrasonic signals to avoid annoying disturbing. Besides, since a speaker and a microphone exist in almost all smart devices, our system could achieve the former requirement. To capture samples properly, we set the sampling rate to 44.1 kHz. Fig. 3 shows the data processing flowchart of MetaDigit.



(a) Digit '0' (b) Digit '1' (c) Digit '2' (d) Digit '3' (e) Digit '4' (f) Digit 5 (g) Digit '6' (h) Digit '7' (i) Digit '8' (j) Digit '9'

(k) Digit '0' (l) Digit '1' (m) Digit '2' (n) Digit '3' (o) Digit '4' (p) Digit '5' (q) Digit '6' (r) Digit '7' (s) Digit '8' (t) Digit '9'

Figure 4: The spectrograms of writing ten basic digits with the device placed vertically (upper) and horizontally (lower)

2.2 Assumptions

- **Motion Continuity**

Based on the observation of users' motions, we assume that users perform a digit continuously, that is, there is no obvious interval during an input motion. Besides, velocities of finger writing are relatively consistent in different parts of a digit motion.

- **Extracted Digits Completeness**

In this paper, we don't focus on how to segment valid motions in the signal stream accurately. Thus, we assume that the frequency shifts caused by the users are properly segmented from the raw audio signals as shown in Fig. 4. Further consideration about removing these assumptions would be in our future work.

2.3 Preprocessing

In the preprocessing stage, we filter the obtained raw acoustic signals and then convert them into time-frequency domain by short-time Fourier transform (STFT), making the data more suitable for deep learning algorithms.

2.3.1 ***Signal Filtering.*** After capturing echoes with a microphone, the raw data are fed into filters to reduce noise and boost SNR. Since the frequencies of target signals vary around 19 kHz, we only need to focus on a certain frequency band of received signals, which not only reduces the interference of out-band components but also decreases the computation overhead in the following steps. Therefore, we make use of a band-pass filter to remove unwanted signal components. To determine filter parameters, we have to quantize the frequency shifts due to the performed gesture. Considering that the finger moves at most 2.6 m/s velocity and the velocity of acoustic signal in the air is 340 m/s, the consequent maximal frequency shift is about 292 Hz (rounded to 300 Hz) which is determined by Doppler effect.

$$\Delta f = f_o \times |1 - \frac{v_s \pm v_t}{v_s \mp v_t}| \tag{1}$$

where f_o, v_s, and v_t represent the frequency of original emitted signals, sound speed in the air, and the gesture velocity, respectively. Based on this, we employ a 6-order Butterworth band-pass filter with a passing band of $[18700, 19300]$ Hz. We then use a 3-order Butterworth band-stop filter to remove the bands near central frequency ($[18985, 19015]$ Hz) to further enhance signals.

2.3.2 ***Signal Transformation.*** After filtering, we transform the one-dimension sequence into two-dimension spectrogram by STFT, in order to reveal the Doppler shifts caused by gestures. In detail, we perform STFT on each signal sequence with the frame length of 8192 data points (corresponding to 0.186 s) and the overlapping length of 7168 data points (corresponding to 0.163 s). These two key parameters are determined by considering the trade-off between time and frequency resolution. The combination of them provides a frequency resolution of 5.38 Hz and a time resolution of 23 ms to track Doppler shifts. We limit the frequency range between $[18700, 19300]$ Hz and then save it as an image. The final input signal data of one user is shown in Fig. 4.

2.4 Few-shot Learning Based Classification

As mentioned above, performance degradation might appear when we test a new user's data which are not in the training set. Further, due to a small amount of testing data, the model tends to overfit the testing data if we try to transfer or retrain on the new user's data.

To mitigate the issue, we are motivated by few-shot learning insights. A typical dataset of a few-shot learning method is divided into two major parts. One is training set, the other is testing set. The training set aims to provide enough prior knowledge, and the testing set provides k shots while we evaluate the model. These k shots are called a support set, which literally means it serves as additional data to support the model to be adaptive to a new user's domain. Thus, typical and notable support k shots are essential with these methods.

To confirm how much the few-shot learning methods outperform the traditional CNN model, we evaluate the following three models, including MobileNet [6], RelationNet [8], MetaSense [5]. The MobileNet is a model designed for mobile devices, which are lack of computing capacity, and speeds up inference on edge devices. RelationNet is an embedding-learning based model, which uses a CNN module as a feature extractor, and is expected to learn the relation scores between support samples and query samples. A higher score of a support class means the test sample is more similar to that class. MetaSense is a MAML-based algorithm [4], which is an adaptive model and would retrain on only a few data instances from a target user and then rapidly adapt to this new user's condition/domain. The experiment and evaluation are discussed in Sec. 3.1 and Sec. 3.2.

2.5 Zero Testing Shot Learning Strategy

Instead of using few-shot learning models which require some new user's samples or using a deeper MobileNet which inferences not really responsible and requires more data to train, we propose to use a LeNet [7] trained with acoustic signals collected from one user under our redesigned scenarios. To further illustrate that, we have done an experiment on a desktop using data from No.10 user in Online Experiments. Done with 10-fold cross-validation but without data augmentation, the comparison of using a simpler model and Mobilenet is shown in Table 1, which indicates that Lenet is more responsible and suitable to the data than Mobilenet while maintaining acceptable accuracy.

Table 1: The comparison of Mobilenet- and Lenet-based models

	Number of parameters		Training Time (s)	
Input Size	Mobilenet	Lenet-based	Mobilenet	Lenet-based
32*32	3239114	62006	66.92	30.76
56*56	3239114	246326	58.11	40.21
84*84	3239114	636086	111.35	74.65
112*112	3239114	1213006	186.85	66.62
140*140	3239114	1980086	277.82	112.67
168*168	3239114	2934326	347.33	198.28
196*196	3239114	4076726	432.85	174.82
224*224	3239114	5407286	484.48	271.79
	Average Accuracy		Inference Time (ms)	
Input Size	Mobilenet	Lenet-based	Mobilenet	Lenet-based
56*56	7.44%	83.88%	22.0	3.6

There are two key insights in data. First, when users get used to performing the digits in a slow manner, due to the intrinsic limits of STFT and doppler effect, the doppler shifts of the input image are too unobvious to be correctly recognized. Second, the subjects are likely to perform the same trajectory after several records because of muscle memory. Even though we collect 100 times per digit from one user, the doppler shifts are similar except the occurrence time in an input image.

Based on the above observations, we redesign data collection scenarios to cover more practical situations of unseen users. To be more specific, we roughly divide the writing area into three scales (*i.e.*, small, normal, large) and three velocities (*i.e.*, slow, normal, fast), that is, there are nine scenarios to collect data. Furthermore, we apply some data augmentation strategy (*i.e.*, time-axis shifts) to further improve the generalization of the model. In this way, our model could predict an unseen user's input in an acceptable accuracy while only trained on one user's data. Noticing that the velocity is the core of Doppler effect, we divide the 10 digits into 30 classes according to three velocity levels. We make a sum of different speeds to make the final classification as Eq.(2) below.

$$C = Argmax(P(i)), \; where \; P(i) = \sum_{j}^{all\ velocity} P(i,j), \qquad (2)$$
$$i \in [0,9] \; and \; j \in \{slow, normal, fast\}$$

where C is the predicted digit and $P(i,j)$ is the probability of digit i with velocity j.

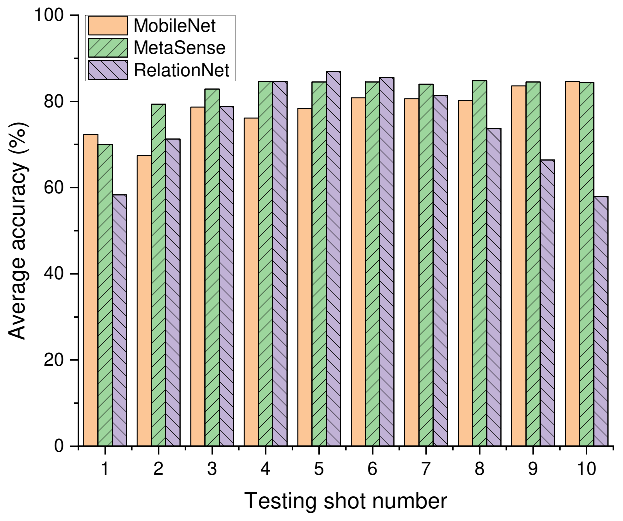


Figure 5: The accuracies over different testing support shots in leave-one-user-out scenarios

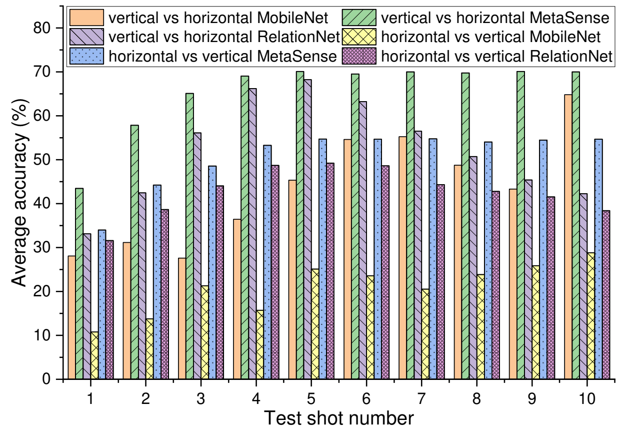


Figure 6: The accuracies over different testing support shots in cross orientations scenarios

3 EXPERIMENTS AND EVALUATION

3.1 Experiment Setup

We implement the prototype on an Android platform (Samsung Galaxy Tab S2) for transmitting and receiving data or even executes inferences in Online Experiments. To relieve the burden of deployment, this project [2] enables android apps to run python codes, which helps to ensure the preprocessing outputs on the desktop and the mobile phones are identical.

1) **Offline Experiments.**
Preliminarily, we implement preprocessing and models on a desktop in this experiment. We conduct the experiment in an indoor and quiet environment. More specifically, we recruit 6 volunteers, including 5 males and 1 female. Our subjects are between 22 and 25 years old and not familiar to perform gestures, so we instruct them to follow the predefined path of all digits without restricting the speed and range (depends on users themselves) of their gestures. Every subject performs 100 times per digit. As a result, we

Table 2: The digits recognition times with vertical placed device.

without multi-labels strategy

User#	0	1	2	3	4	5	6	7	8	9	Avg
No.2	3	0	2	3	6	10	3	1	8	7	4.3
No.10*	9	8	8	10	5	10	5	6	8	8	7.8
Avg	6	5	2	6.5	6.5	10	3	4	9	8.5	6.1

with multi-labels strategy

User#	0	1	2	3	4	5	6	7	8	9	Avg
No.1	9	6	7	10	8	9	8	10	10	10	8.7
No.2	8	7	8	9	9	10	6	10	9	10	8.6
No.3	9	9	9	10	7	9	7	10	6	9	8.5
No.4	9	10	9	9	9	7	7	10	6	9	8.5
No.5	10	8	6	8	10	7	9	10	6	10	8.4
No.6	8	10	9	7	8	8	8	10	4/**8**	9	8.1/8.5
No.7	10	7	10	10	10	5/**10**	9	10	8	10	8.9/9.4
No.8	10	10	7	5/**8**	10	7	10	10	10	10	8.9/9.2
No.9	9	10	9	8	10	10	5	10	10	8	8.9
No.10*	9	10	6	10	9	10	10	8	10	10	**9.2**
No.11	5/**10**	10	10	10	9	10	10	9	9	9	9.1/9.6
No.12	7	8	8	10	10	10	4/**9**	10	10	9	8.6/9.1
Avg	8.6	8.8	8.2	8.8	9.1	8.5	7.8	9.8	8.2	9.4	8.7
Avg /	9	8.8	8.2	9.1	9.1	8.9	8.2	9.8	8.5	9.4	8.9

*This user provides the training data of nine scenarios.

obtain a dataset with 6000 digits samples. More importantly, data are collected from 2 different device orientations of the device (horizontal from 4 subjects and vertical from 2 subjects), which means our dataset contains 6 different users' domains with 2 different orientation domains.

2) **Online Experiments.**
First, we collect data from a user and then train our model on the data. Then, we implement preprocessing and models on a real-world device using TensorFlow [1] in this experiment, getting metrics directly on prototype with all the users. To be more specific, as mentioned in Sec. 2.5, we recruit one user (denoted as No.10) to perform 5 times per digit under nine different scenarios with vertically and horizontally placed devices separately. As a result, we obtain a dataset with 900 digits while 450 in vertical and 450 in horizontal placements respectively, which is smaller but more general than the dataset of Offline Experiments.

3.2 Evaluation

In this work, we focus on how to reduce the samples from new users and mitigate performance degradation while transferring to new users. Therefore, We choose to test three scenarios, including offline evaluation (part.1 and part.2) and online evaluation (part.3):

1) **Leave-One-User-Out Scenario.** We train on 5 users' data and test on the other user's data, evaluating the few-shot methods and MobileNet. We set the training shots number to 5 and 2 in RelationNet and MetaSense in the training phrase respectively. After the model is trained, we test the accuracy with different numbers of testing support shots. Fig. 5 indicates that MobileNet performs better as the test shot number increases. MetaSense converges rapidly after several testing support shots while RelationNet is sensitive to the number of training shots, that is, RelationNet performs well around 5 testing support shots. Training data are mixed with the testing support set while training MobileNet. If MobileNet tunes parameters directly on target support set, it would achieve better performance just like in previous work [5]. In this scenario, few-shot methods don't essentially outperform MobileNet with different testing support shot numbers, especially when the testing shot number is 1 or 10. The results indicate that

Table 3: The digits recognition times with horizontally placed device.

without multi-labels strategy

User#	0	1	2	3	4	5	6	7	8	9	Avg
No.2	6	9	8	10	10	10	5	9	10	10	8.7
No.10*	8	5	9	10	9	10	6	10	10	9	8.6
Avg.	7	7	8.5	10	9.5	10	5.5	9.5	10	9.5	8.65

*This user provides the training data of nine scenarios.

the trained few-shot methods suit for several testing support samples rather than larger testing support shot numbers, which motivates us to improve the data quality based on internal physics properties rather than using sophisticated models which are hard to deploy.

2) **Cross-Orientation Scenarios.** We train on one orientation (*e.g.*, vertical) and test on the other orientation (*e.g.*, horizontal). Fig. 6 shows that while transferring to a new orientation scenario, few-shot based algorithms perform better in several testing support shots, especially for MetaSense. However, the results are not acceptable while testing under cross-orientation scenarios. The reasons could be visually shown in Fig. 4. The distributions of digits between two orientations are quite different, which means if we train models purely on data of one orientation and test with the other orientation, the system can't work properly. Thus, providing balanced and just enough scenarios data to the model would be a more promising way to mitigate this challenge, which would be in our future work.

3) **Real Deployment Usage Scenarios.** We train the model with No.12 user and then deploy the model directly to the device, and then recruit 12 users (including the training user) to perform 10 times per digit with the deployed device. The results are shown in Table 2. Due to the unfamiliarity of performing digits, we ask the users whether the recognition accuracy is acceptable (lower than 6). If so, they perform another 10 times to confirm they are familiar with digits input (bold data after the slash). What's more, we test on the training user with different classification strategies. Results indicate that our strategy improves accuracy. The overall accuracy of the users, whose data are not in the training set, achieves over 85%. Compared with few-shot learning, our strategy requires zero support shot from new users, which relieves the data collecting effort from unseen users.
As shown in the upper part of Fig. 4, due to the similar patterns of frequency shifts, some digits would get misclassified easily. For example, if we want to input a digit '6', we may get a classification of digit '0' because of the common parts of frequency shifts. Similarly, digits '2' and '8' have slight degradations on classification accuracy.

In general, few-shot learning methods provide some insights to reduce the user's training effort and mitigate the cross-orientation degradation issue, utilizing only a few testing support shots, but hard to deploy. To be specific, some methods like meta-learning require "retraining" to some extent. Some methods like few-shot learning require users to provide some kind of support set. However, the distribution is quite different to transfer. We propose a strategy based on the intrinsic observation of data, which leverages zero-shot of unseen users. The strategy could be applied to the horizontally placed device scenario, but transferring to other orientation and still maintaining the accuracy remains a challenge. The mixed-orientation scenarios and full-English letters input scenarios would be in our future work.

4 DISCUSSION AND OPEN ISSUES

4.1 The Placement Orientation

In this work, we have considered the cross-user usage scenarios, in which various writing patterns of different users are taken into account. However, besides this, the cross-orientation problem has also be considered since the placement orientation of a device may vary during usage. We have done a preliminary experiment about extending our method to the other orientation. As shown in Table 3, the accuracies are exceeding 85% with nine scenarios strategy but without multi-label strategy. However, the challenge becomes how to maintain the ability to classify original orientation while transferring to target orientation successfully. As shown in Fig. 4, different orientations come with different distributions. Thus, it would be a challenging part of our future work.

4.2 English Letters Input

In this work, we only focus on studying the challenges of pushing a prototype of digits input system into practice. However, this is far from the goal of designing a practical texts-input system, since English texts have not been taken into account yet. To accomplish this, we plan to extend our method to recognizing basic English letters first, and then combine with language models to infer words and sentences. Nevertheless, this is not a straightforward task as expected. An obvious challenge is that since the number of basic letters is much more than that of basic digits, the problem of pattern ambiguity is probably more serious. That is, the Doppler shifts patterns of basic letters share more similarity, which makes it more challenging to recognize them with high accuracy. Our following work is to first apply our strategy to recognize letters and investigate the possible problems. A good point about English texts recognition is that we can introduce word-formation rules and language models to perform error correction to boost recognition accuracy.

5 CONCLUSION

Motivated by designing a practical digit-input system, we have carefully explored the shortcomings of existing machine learning-based schemes in terms of six different metrics including recognition accuracy, response time, training data and time overhead, model training time, and user overhead. Although existing methods show distinctive performance in certain metrics especially accuracy, the shortcomings in other aspects hinder their deployment in real-world scenarios. Consequently, we make an attempt towards the goal by designing MetaDigit which can recognize all digits with acceptable accuracies in real time, with nearly zero effort from a new user.

ACKNOWLEDGMENTS

This research was supported in part by the China NSFC Grant (61802264, 61872248), Shenzhen Science and Technology Foundation (No. JCYJ20180305124807337, No.ZDSYS20190902092853047), Guangdong Science and Technology Foundation (2017A030312008, 2019B111103001, 2019B020209001), GDUPS (2015), Natural Science Foundation of SZU (No. 860-000002110537). Yongpan Zou is the corresponding author.

REFERENCES

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*. 265–283.

[2] Chaquo. 2020. chaquopy. https://chaquo.com/chaquopy/.

[3] Thomas Deselaers, Daniel Keysers, Jan Hosang, and Henry /A/ Rowley. 2015. GyroPen: Gyroscopes for Pen-Input With Mobile Phones. *IEEE Transactions on Human Machine Systems* 45, 2 (2015), 263–271.

[4] Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. In *ICML*.

[5] Taesik Gong, Yeonsu Kim, Jinwoo Shin, and Sung-Ju Lee. 2019. MetaSense: few-shot adaptation to untrained conditions in deep mobile sensing. In *Proceedings of the 17th Conference on Embedded Networked Sensor Systems*. 110–123.

[6] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. 2017. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861* (2017).

[7] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324.

[8] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. 2018. Learning to compare: Relation network for few-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1199–1208.

[9] Jue Wang, Deepak Vasisht, and Dina Katabi. 2014. RF-IDraw: virtual touch screen in the air using RF signals. *ACM SIGCOMM Computer Communication Review* 44, 4 (2014), 235–246.

[10] Yaqing Wang, Quanming Yao, James T Kwok, and Lionel M Ni. 2019. Generalizing from a few examples: A survey on few-shot learning. *ACM Computing Surveys (CSUR)* (2019).

[11] Yongpan Zou, Qiang Yang, Yetong Han, Dan Wang, Jiannong Cao, and Kaishun Wu. 2019. Acoudigits: Enabling users to input digits in the air. In *2019 IEEE International Conference on Pervasive Computing and Communications (PerCom*. IEEE, 1–9.