Phone-based CSI Hand Gesture Recognition with Lightweight Image-Classification Model

Ashkan Arabi
the University of Texas at El Paso
El Paso, TX, United States
aarabimian@miners.utep.edu

Michael Straus
Columbia University
New York, NY, United States
mjs2435@columbia.edu

Zijie Tang
Temple University
Philadelphia, PA, United States
zijie.tang@temple.edu

Zhengkun Ye
Temple University
Philadelphia, PA, United States
zhengkun.ye@temple.edu

Yan Wang
Temple University
Philadelphia, PA, United States
y.wang@temple.edu

ABSTRACT

As years pass, smartphones are becoming a larger part of daily lives, causing users to interact with them more than ever. There are moments, however, when it becomes difficult for the user to operate their device directly. Currently, a user can either touch their devices for direct interaction, or use voice commands for simpler tasks. Although these two methods are very capable means of interacting with the devices, they have their limitations. Touching a physical device is not always practical, while voice commands become ineffective in loud environments. A good example would be if the user is washing dishes in a noisy environment, where neither physical control nor voice commands are convenient. Existing systems of smartphone CSI gesture recognition rely on manual feature extraction which could be hard to implement as gestures grow in number and complexity. We study the feasibility of using lightweight image classification models with minimal preprocessing by implementing and testing the performance of such an architecture. We collect data for five gestures from three setups and two phones, on which our system is able to obtain 90.0% accuracy. Additionally, we investigate the impact of different people, distances, and phones on the system's performance.

CCS CONCEPTS

• **Human-centered computing** → **Ubiquitous and mobile computing**.

KEYWORDS

Wi-Fi Sensing; Gesture Recognition; Channel State Information; Human Computer Interaction

ACM Reference Format:
Ashkan Arabi, Michael Straus, Zijie Tang, Zhengkun Ye, and Yan Wang. 2023. Phone-based CSI Hand Gesture Recognition with Lightweight Image-Classification Model. In *The Twenty-fourth International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing (MobiHoc '23), October 23–26, 2023, Washington, DC, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3565287.3617613

MobiHoc '23, October 23–26, 2023, Washington, DC, USA

ACM ISBN 978-1-4503-9926-5/23/10…$15.00
https://doi.org/10.1145/3565287.3617613

1 INTRODUCTION

As computer hardware and software develop, an increasing portion of our daily lives enters the digital realm. To maintain the same levels of productivity on such interfaces, many researchers have devoted themselves to making human-computer interaction more natural, especially with smartphones.

There are many situations where a user may not be able to directly operate their smartphone. In such a scenario, the most prominent alternative methods of interacting with smartphones are Bluetooth devices (e.g. AirPods) and voice assistants (e.g. Siri); both of which have their limitations. While Bluetooth devices are a reliable and easy-to-use way to interact with smartphones, they require the user to wear them beforehand. Additionally, they still require people to touch them, which is not convenient in certain situations. Likewise, voice assistants are another intuitive method of smartphone interaction, but they become ineffective in noisy environments.

Hand gestures have the opportunity to cover some of these blind spots. Urgent actions, like answering/declining calls or waking the phone to glance at notifications can be performed quickly with gestures, while also being intuitive to use. If such a technology is successfully integrated, the user experience of smartphones and many other devices could be improved considerably.

Current hand gesture recognition techniques include using cameras [5], wearable devices [15], radar [10], Wi-Fi Received Signal Strength (RSSI), and Wi-Fi Channel State Information (CSI). Vision-based methods have proven to be quite accurate [7], but they suffer from issues such as privacy and quality of the camera. Not to mention that the gesture has to be in its field of view, a very unlikely scenario on a smartphone. Wearable devices tend to also be very accurate, but they are highly intrusive due to their reliance on specialized equipment [15]. Radar devices could be applied, but this solution requires hardware installations and can be costly [10]. RSSI-based systems can rely on the existing Wi-Fi infrastructure in many buildings, but their scope is limited to coarse-grain activities [1]. Current studies using RSSI are only able to reliably detect motions on the order of an arm and require equipment beyond

a smartphone, putting scalable detection of hand gestures out of reach [3].

CSI is a better alternative to RSSI because it contains multiple magnitude channels, and thus more detailed data. It has been extensively studied for gesture detection on many devices, including smartphones. Li et al. [12] use feature engineering for gesture classification on a Nexus 5 with high accuracy. They rely on threshold-based algorithms, which can be difficult to adjust for more complex gestures [2]. To solve this issue, we investigate the use of deep learning, and specifically, lightweight image classification techniques to bypass the feature engineering process. Our work is inspired by Bu et al. [6], who successfully use a large image-classification model for gesture classification on commercial APs. Our architecture, however, is much lighter, making it suitable for deployment on mobile phones. Our main contributions are:

- To the best of our knowledge, this is the first work to employ deep image classification models for CSI hand gesture detection on smartphones.
- We achieve an accuracy of 90.0% in classifying 5 gestures using combined data from 3 setups and 2 smartphones.
- We investigate four impact factors on our model's performance: Smartphone & hand distance, AP & hand distance, different phones, and different users.

2 RELATED WORK

Approaches for gesture recognition can be divided into three main groups: worn sensor-based, camera-based, and radio-frequency (RF)-based.

Worn / Carried Sensor-Based. Different types of worn sensors have been used to facilitate gesture detection for Human Computer Interaction (HCI). Zhang, et al. [17] use a strain sensor in a smart glove to accurately map the shape of the hand. There has also been interest in using ultrasound to detect gestures in challenging environments. Most notably, EchoFlex [14] is able to achieve high accuracy on 10 gestures by strapping an ultrasound transducer to the forearm. While such approaches show great accuracy, they are not easy to carry, and their adaptability is limited due to price.

Vision Based. Vision-based methods have experienced many advancements in recent years due to breakthroughs in image recognition techniques, making them suitable for tasks such as sign-language translation [4]. For example, Kim et al. [9] use an Xbox Kinect sensor to combine vision and depth info for an HCI application. While this and similar systems perform well in complex scenarios, the limited LOS (line of sight) of the sensors and privacy concerns make them unsuitable for deployment on smartphones.

RF Based. RF-based technologies can be categorized into radar, RSSI, and CSI-based methods. Radar gesture detection was seen on Google Pixel 4 smartphones using Google's Soli [13] millimeter-wave radar sensor. It could be used to perform tasks such as skipping songs and muting notifications. However, there are countless phones without that specialized sensor, limiting its adaptability.

Unlike Radar, most devices are equipped with Wi-Fi chips that can be used for gesture recognition. WiGest [1] uses pre-existing computers and access points (APs) to perform general device interaction functions. Their system is resilient to environment or position variations. However, the system uses less detailed RSSI, and thus is plagued with issues of distinguishing identical gestures across different people. The lack of applicability when it comes to similar situations hinders its large-scale deployment.

As opposed to RSSI, CSI has proven to be much more flexible. It contains Channel Frequency Response information of each Wi-Fi subcarrier, which changes based on objects in the receiver's environment [4]. Historically, CSI has only been available internally for the chips [8]. The release of custom firmware tools like Nexmon [8], gave researchers the tools to experiment with this data. For example, WiGER [3] uses commercial APs, and can operate with multiple objects obstructing its view, even when the user is not between the transmitter and receiver. However, CSI captured on such devices has a much higher resolution compared to smartphone Wi-Fi chips. They use high packet transmission rates of 100Hz, which are not accessible on all mobile devices. Similarly, Li et al. achieve very high accuracy by using techniques such as dynamic time warping (DTW) and various filters on the CSI data of a Nexus 5 phone [12] for the task. That said, manual feature extraction is difficult to implement as gestures grow in number and complexity [4].

Bu et al. [6] tackle this issue by using a VGG image-classification model pretrained on the ImageNet dataset. They then fine-tune their model for CSI gestures, and achieve an almost perfect accuracy. Their model's main drawback is the very large size of more than 13 million convolutional parameters, which prevents it from future smartphone implementation. Inspired by their paper, we attempt to create a much lighter model to perform the same task on a smartphone.

3 BACKGROUND & CHALLENGES

This section covers background information about technologies and tools used for this paper. Additionally, we'll cover challenges specific to using smartphones for gesture classification as opposed to traditional NICs.

3.1 CSI and CSI Tools

Channel State Information (CSI). Commercial off-the-shelf (COTS) devices adhering to 802.11n standards use Orthogonal Frequency division Multiplexing (OFDM) to achieve high data rates and reduce Bit Error Rate. OFDM extracts the channel frequency response in the format of CSI which can give the device information about the stability of Wi-Fi subcarries for data transmission. For each subcarrier, the data is in the form of a magnitude and a phase in the form of a complex number [2].

Nexmon CSI Tool. Nexmon [8] is a C-based Wi-Fi chip patching tool that allows us to modify the firmware of our Nexus 5 smartphones. They offer a CSI extractor patch that we use for data collection. Nexmon firmware can access low-level transmission information, and saves the CSI of each subcarrier (64 in our case) as complex pairs in UDP frames. Groups of UDP packets are then stored in PCAP files which are then processed in our pipeline.

3.2 Challenges

Compared to using a Network Interface Controller (NIC), there are several challenges that arise when using smartphones as the source of CSI data.

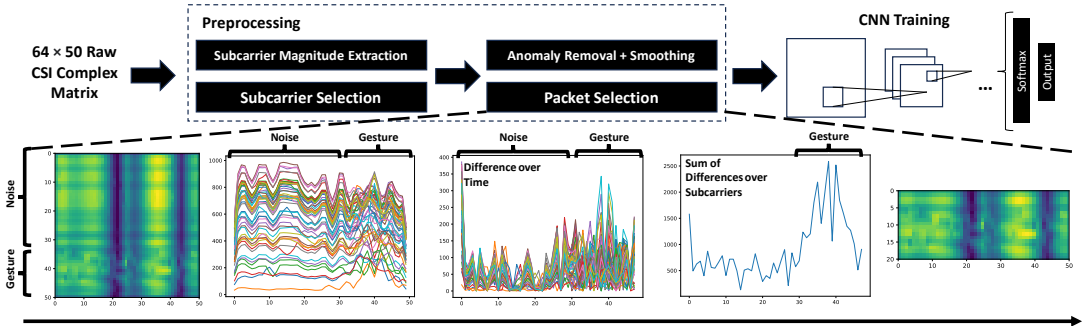


Figure 1: System overview

Software Limitations. Commercial Wi-Fi chips, like those in smartphones, do not make CSI available for external usage. [8] This data can only be obtained by using custom firmware on a number of supported devices. We use LG Nexus 5s along with Nexmon for this exact reason.

Lower Sampling Rate. Smartphones, due to power constraints, cannot collect samples with the same high frequency as NICs. This can lead to lower-resolution data that can negatively impact model performance. We combat this by keeping as many subcarriers in our preprocessing as possible.

Processing Power. On smartphones it is crucial for all software to be as lightweight as possible due to their limited computing ability. While large image classification models such as those used in [6] are very accurate, they are not feasible for smartphones. We address this problem by minimizing our parameter count.

4 SYSTEM DESIGN

Our system works on the idea that fluctuations in the magnitude of subcarriers over time can be represented as a matrix, therefore can be seen as an image-classification problem. We keep preprocessing minimal and leave the task of feature extraction to the model.

4.1 System Overview

A summary of the system overview can be seen in Figure 1. After collecting the CSI data using the Nexmon tool, the raw CSI data is a complex matrix with dimensions 64×50, where 64 is the number of subcarriers and 50 is the packet count. During preprocessing, we discard phase information because no change is seen in response to gestures. Then, we drop 14 static or noisy subcarriers of our original 64. This is followed by anomaly removal and smoothing, before pruning the packages into 20 for training. The trimming window was set to 20 because participants were required to keep gestures under two seconds while we recorded the CSI at 10 Hz.

The resulting 20 by 50 magnitude matrix is directly used as training/testing data, as it can be treated like visual input for an image-classification CNN model. It will recognize the patterns seen over time in each subcarrier's magnitude caused by activity.

4.2 Pre-processing

Raw data, as captured from each Wi-Fi packet through the Nexmon utility [8], is a complex array of size 64. We use a 20MHz channel width and get 54 valid subcarriers, discarding the other 10 due to them being static. This complex array can be decomposed into magnitude and phase arrays. As shown in Figure 1, phase is discarded as it shows no change when gestures are performed. We also discard another 4 of the 54 original subcarriers due to excessive noise.

Our observations show that in the remaining 50 subcarriers, there is occasional noise. To tackle this, we first remove any outliers based on standard deviation and replace them with the median of subcarrier signal strength with respect to time. Next, we apply a Gaussian filter with a window size of 3 to further bring random variation down to a manageable level.

If we capture multiple packets and stack their magnitude arrays over time, we get a 50 by x matrix, where x is the number of captured packets. We record 50 samples (5 seconds) for each sample to give participants enough time to perform the gestures, which gives us an initial 50 by 50 matrix. We ensured that all participants finished any gesture in 2 seconds so that we could then select only the 20 packets containing movement data from the 50 we have collected for each sample. This results in a final 20 by 50 matrix that is fed into the model. The pruning process is as follows.

Firstly, a rolling difference operation is applied along the time axis of the CSI matrix. This operation calculates the absolute difference between consecutive subcarrier values within a sliding window of size two. The purpose of this step is to capture variations or changes in the subcarriers' magnitudes, which can be indicative of important features in the CSI.

Next, a rolling difference operation is performed along the subcarrier axis of the modified CSI matrix. This time, the rolling difference is calculated between consecutive packets within a sliding window of size two. The intention here is to capture differences between neighboring packets, which can show if their rates of change are synchronous or not.

Subsequently, a sum operation is executed on the resulting matrix from the previous step that returns a time series. At a given point, if this sum is a small number, it means that most of the subcarriers change synchronously, indicating that the current data is background noise. If this sum is large, it indicates hand gesture activity.

Finally, we employ a rolling sum with a window of size equal to the desired packet count, which is 20, to check the interval where the resulting array from the previous operation is maximized. The starting index of that window at its maximum point will be the same as the starting index of the desired gesture. The ending index

Layer	size	activation
Batch Normalization	–	–
Conv2D	$3 \times 3 \times 16$	relu
Conv2D	$3 \times 3 \times 16$	relu
Max Pooling	2×2	–
Conv2D	$3 \times 3 \times 32$	relu
Conv2D	$3 \times 3 \times 32$	relu
Max Pooling	2×2	–
Conv2D	$3 \times 3 \times 64$	relu
Conv2D	$3 \times 3 \times 64$	relu
Max Pooling	2×2	–
Flatten	–	–
Dense	256	relu
Dropout	0.5	–
Dense	256	relu
Dropout	0.5	–
Dense	5	softmax

Table 1: Model architecture

is obtained by adding, in our case, 20 to the starting index. These two indices are then used to crop the 50 by 50 matrix into 20 by 50, leaving us with the final result of preprocessing.

We use this method because we occasionally observed magnitude fluctuations due to noise. Methods such as low-pass filtering could not be employed due to the low sampling-rate. Observations show that if the fluctuations are synchronous across subcarriers, they are noise, but if they are not coordinated they are caused by the hand gesture. This technique was able to correctly prune all samples compared to ground truth.

4.3 Classification Model

Inspired by Bu et al.'s work [6], we employ a convolutional image-classification model to detect the given gesture based on the magnitude matrix of each sample. Our model, however, is much lighter-weight to make it feasible for deployment on smartphones. While said paper uses pre-trained VGG-16 and VGG-19 models, with more than 13 million parameters just on the convolutional layers [16], our goal was to have a much lower parameter-count to make it possible to deploy on smartphones.

Our model can be described as a non-linear mapping function $\mathcal{F}_\omega(\cdot)$, where ω represents the model weights. The training process is as follows:

$$\arg\min_{\omega} \sum_{i=1}^{\mathcal{N}} \mathcal{L}\left(\mathcal{F}_\omega\left(x_i\right), y_i\right) \quad \text{s.t. } x_i \in \mathbb{X}, y_i \in \mathbb{Y}, i = 1, ..., \mathcal{N}, \tag{1}$$

where $\mathcal{L}$ represents crossentropy loss, $\mathcal{N}$ is the number of training examples, and x_i and y_i represent the i^{th} training data and the respective label. Each x_i is an $m \times n$ matrix, where m is the number of packets, and n represents the number of subcarriers. In our case, m and n are 20 and 50 respectively.

Our proposed model is shown in Table 1, containing 8 hidden layers; six of which are convolutional, and 2 are dense. For each convolutional layer, we use a filter size of 3 and preserve the dimensions. The output of each pair of convolutional layers is down-sampled using a 2x2 max-pooling operation (3 in total). Then, the output is flattened and fed into two dense layers of size 256, followed by the final dense layer of size 5. The final model has less than 350,000 total parameters.

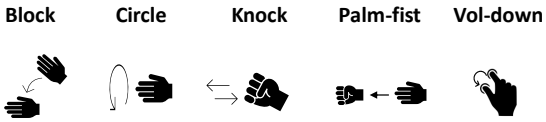


Figure 2: The five gestures identified in our model.

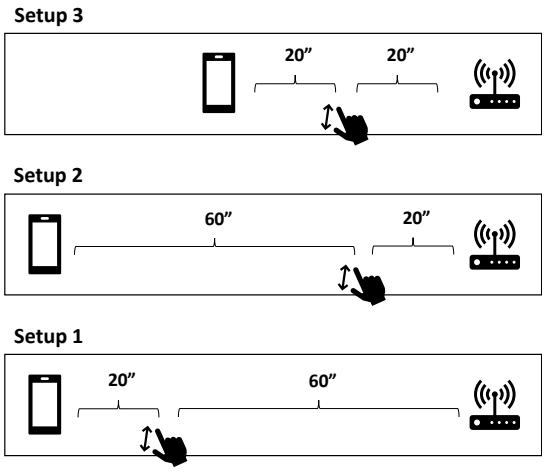
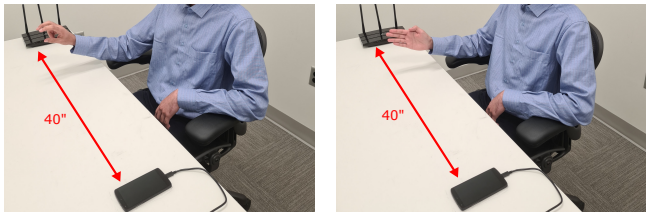


Figure 3: Three setups for data collection.



(a) Performing "voldown" **(b) Performing "block"**

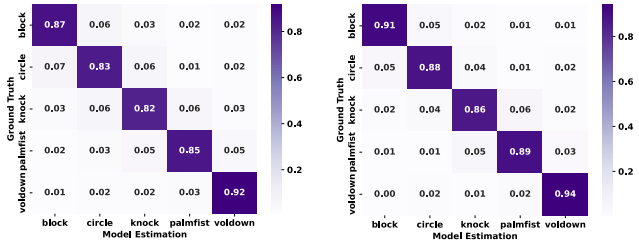
Figure 4: The experiment environment

5 EXPERIMENT AND EVALUATION

We evaluate the performance by collecting CSI data from four people, three setups and two phones. We first measure the model's overall performance using data from one user in setups 1,2,3 and using both phones. Then we measure the effect of four impact factors: 1) usage on different phones, 2) different users, 3) proximity of access point and gesture, 4) and proximity of phone and gesture.

5.1 Experimental Methodology

Data Collection. We use two Nexus 5 Android smartphones with stock hardware, along with a TP-Link AC1750 router for this experiment. Both phones run Android 6.0.1 and have been rooted to allow for the installation of custom "Nexmon CSI" firmware [8]. The sampling rate on the phones is 10Hz, which is considerably lower than dedicated APs that are able to capture CSI at rates such



(a) 4 people in one scenario **(b) Person 1 in all scenarios**

Figure 5: Overall confusion matrices

as 100Hz [11]. As shown in Figure 2, we design five gestures which are as follows: block, circle, knock, palm-fist, and vol-down.

We use an efficient scheme to reduce the time and resources needed for data collection while still studying all of our impact factors. Person 1 performs two datasets for all gestures in all environments using both phones, totaling 300 samples per gesture (12 datasets of 25 per gesture). Persons 2, 3, and 4 each will perform each gesture 50 times (two datasets of 25) in environment 3 using phone 1.

We conduct the data collection in a controlled, noise-free room with minimal activity and rich multipath reflections, which is shown in Figure 4. The smartphone and router are placed on both sides of the table, and subjects perform gestures directly in the line of sight of the phone and router. Each time, the participants are given five seconds (50 packets) to perform the gesture. These datasets can be found on Kaggle[1].

Experimental Setup. We found the model to generalize best at 150 epochs with a batch size of 64. To make the learning more stable with such a large number of epochs, we use learning rate (LR) decay. We initialize the model with a learning rate of 0.003 in the Adam optimizer, and for every 10 epochs, multiply the existing learning rate by a drop rate of 0.8. For example, if the current learning rate is 0.001, after 10 epochs it will be 0.0008, and 0.00064 in the 10 epochs that follow. The code can be fond on GitHub [2].

Evaluation Metrics We define the following two metrics to evaluate the performance: **1) *Accuracy.*** The model's accuracy is measured as the total number of correct observations divided by the number of all observations. **2) *F1 Score.*** The F1 score of combines the precision and recall of the class, and is less prone to bias if any of the gestures in the datasets are unbalanced. It is defined as follows:

$$F_1 = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{2}$$

where precision and recall are defined as:

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives} \tag{3}$$

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives} \tag{4}$$

It must be noted that each of these metrics are calculated for one of the five classes. Any reference to these metrics for a complete dataset refers to the mean from all five gestures.

[1]https://www.kaggle.com/datasets/ashkanarabi/nexus-5-csi-hand-gestures
[2]https://github.com/AshkanArabim/phone-gesture-recognition

All tests are performed using 5-fold cross-validation to minimize effects caused by the data distribution. The final reported result represents the mean accuracy and F1 scores of all 5 folds. Note that in comparison studies, unless otherwise specified, the model is trained and tested on each condition in isolation from other scenarios.

5.2 Overall Performance

In order to efficiently assess the overall performance of our model, we devised a streamlined evaluation approach that allowed us to save time and resources. Instead of having all four individuals perform the gestures using both smartphones in all three setups, we strategically divided the evaluation process into two phases. (Figure 5)

Firstly, we trained and tested the model using data solely from person 1, encompassing all three setups and both phones. This approach provided us with a commendable 90.0% accuracy with an F1 score of 0.898. This demonstrates the model's robustness to different phones of the same model and different distances. Then, we sought to validate the model's performance across a more diverse dataset. To achieve this, we used data from all four individuals but limited the evaluation to a specific scenario (setup 3, phone 1). The results showed an accuracy of 86.0% and an F1 score of 0.862, showing strong performance when faced with different gesture styles.

Overall, we can see that our model can obtain a good accuracy when trained on smartphone CSI. This is true for training on data from different people and from different setups.

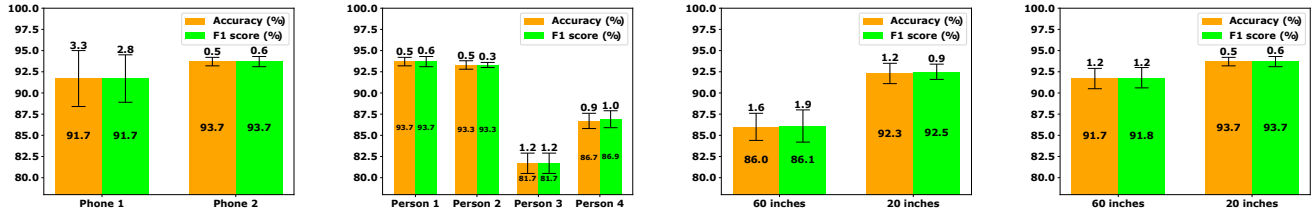
5.3 Impact Factors

Impact of Different Phones. For this experiment, we use data from person 1 performing the five gestures in setup 1, 2, and 3 using each phone (Figure 6a). The model gets 91.7% and 93.7% for the performance of phone 1 and 2 respectively, and F1 scores of 0.937 and 0.917. The difference in performance could have been caused by slight environmental variations, but they can also hint that the slight differences in phones' Wi-Fi chips, even if they are the same model, can have an effect on CSI measurements. This could be a limit for large-scale implementation.

Impact of Different Users. We measure the performance of the model on four users, asking them to perform the five gestures in Setup 3 with phone 1. A moderator was present to ensure correct posture and participants' hand form. The results show a significant difference in the performance of participants, ranging from 81.7% to 93.7% as seen in Figure 6b.

Impact of Access Point Proximity. We train the same model on data from Setup 1 (phone/hand 20" apart) and 3 (60" apart) using data from User 1 on Phone 1. We observe from the results (Figure 6c) that there is a significant difference in performance when increasing the distance of AP from 20" to 60" (92.3% and 86.0% respectively). Our model performs noticeably better when the AP is placed closer to the hand.

Impact of Smartphone Proximity. In order to investigate this factor, we repeat the same process with Setups 2 and 3. We observe a drop in performance when the phone is further from the hand in Figure 6d. However, the difference is less significant than for AP



(a) Two different Nexus 5s (b) Four different people (c) Impact of AP proximity (d) Impact of phone proximity

Figure 6: Results from all impact factors

distance, as we achieved 93.7% and 91.7% accuracy when the phone was 20" and 60" from our hands. The selected metrics are therefore not seriously affected by the proximity of one's mobile device.

6 DISCUSSION

Limitations. This is by no means a comprehensive model, and it has several limitations.

- Firstly, since all of our experiments were done on Nexus 5 smartphones, we cannot make any assumptions about the performance of this system on other smartphone hardware.
- Secondly, experiments were done in a controlled room, so our results do not necessarily translate to other environments.

Future Work. Our future work will consider addressing the above problems as top priority by expanding the study to more smartphones and environments, while also developing more robust preprocessing and classification to handle these scenarios. Additionally, the system will benefit from a binary noise vs gesture classifier for real-word deployment. We would also like to experiment with scenarios where the hand is not directly between the smartphone and AP.

7 CONCLUSION

In this paper, we explore hand gesture recognition using Channel State Information (CSI) collected by smartphones, enabling users to perform quick phone operations when in situations such as washing dishes in a noisy environment. We examine the feasibility of a lightweight image-classification convolutional neural network (CNN) to detect hand gestures with minimal feature extraction, which can adapt to different scenarios due to not needing manual feature engineering. By designing five hand gestures and collecting data from three setups, two phones and four people, our approach achieves an average of 90.0% accuracy. Furthermore, we conduct a comprehensive study on the impact factors, demonstrating the scalability of our approach to different users, different distances from users & phones and distances from users & APs, and different phones.

8 ACKNOWLEDGEMENTS

This research includes calculations carried out on HPC resources supported in part by the National Science Foundation through major research instrumentation grant number 1625061 and by the US Army Research Laboratory under contract number W911NF-16-2-0189. This work was supported in part by the US National Science Foundation Grant CNS 2150152.

REFERENCES

[1] Abdelnasser, H., Youssef, M., and Harras, K. A. Wigest: A ubiquitous wifi-based gesture recognition system. In *2015 IEEE Conference on Computer Communications (INFOCOM)* (2015), pp. 1472–1480.

[2] Ahmed, H. F. T., Ahmad, H., and Aravind, C. Device free human gesture recognition using wi-fi csi: A survey. *Engineering Applications of Artificial Intelligence 87* (2020), 103281.

[3] Al-qaness, M. A. A., and Li, F. Wiger: Wifi-based gesture recognition system. *ISPRS International Journal of Geo-Information 5*, 6 (2016).

[4] Al-Shamayleh, A. S., Ahmad, R., Abushariah, M. A., Alam, K. A., and Jomhari, N. A systematic literature review on vision based gesture recognition techniques. *Multimedia Tools and Applications 77* (2018), 28121–28184.

[5] Alsheakhali, M., Skaik, A., Aldahdouh, M., and Alhelou, M. Hand gesture recognition system. *Information & Communication Systems 132* (2011).

[6] Bu, Q., Yang, G., Ming, X., Zhang, T., Feng, J., and Zhang, J. Deep transfer learning for gesture recognition with wifi signals. *Personal and Ubiquitous Computing* (2020), 1–12.

[7] Chin-Shyurng, F., Lee, S.-E., and Wu, M.-L. Real-time musical conducting gesture recognition based on a dynamic time warping classifier using a single-depth camera. *Applied Sciences 9*, 3 (2019).

[8] Gringoli, F., Schulz, M., Link, J., and Hollick, M. Free your csi: A channel state information extraction platform for modern wi-fi chipsets. In *Proceedings of the 13th International Workshop on Wireless Network Testbeds, Experimental Evaluation & Characterization* (New York, NY, USA, 2019), WiNTECH '19, Association for Computing Machinery, p. 21–28.

[9] Kim, K., Kim, J., Choi, J., Kim, J., and Lee, S. Depth camera-based 3d hand gesture controls with immersive tactile feedback for natural mid-air gesture interactions. *Sensors 15*, 1 (2015), 1022–1046.

[10] Kim, Y., and Toomajian, B. Application of doppler radar for the recognition of hand gestures using optimized deep convolutional neural networks. In *2017 11th European Conference on Antennas and Propagation (EUCAP)* (2017).

[11] Kresge, K., Martino, S., Zhao, T., and Wang, Y. Wifi-based contactless gesture recognition using lightweight cnn. In *2021 IEEE 18th International Conference on Mobile Ad Hoc and Smart Systems (MASS)* (2021), IEEE, pp. 645–650.

[12] Li, T., Shi, C., Li, P., and Chen, P. A novel gesture recognition system based on csi extracted from a smartphone with nexmon firmware. *Sensors 21*, 1 (2021).

[13] Lien, J., Gillian, N., Karagozler, M. E., Amihood, P., Schwesig, C., Olson, E., Raja, H., and Poupyrev, I. Soli: Ubiquitous gesture sensing with millimeter wave radar. *ACM Transactions on Graphics (TOG) 35*, 4 (2016), 1–19.

[14] McIntosh, J., Marzo, A., Fraser, M., and Phillips, C. Echoflex: Hand gesture recognition using ultrasound imaging. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 2017), CHI '17, Association for Computing Machinery, p. 1923–1934.

[15] Shukor, A. Z., Miskon, M. F., Jamaluddin, M. H., bin Ali@Ibrahim, F., Asyraf, M. F., and bin Bahar, M. B. A new data glove approach for malaysian sign language detection. *Procedia Computer Science 76* (2015), 60–67. 2015 IEEE International Symposium on Robotics and Intelligent Sensors (IEEE IRIS2015).

[16] Simonyan, K., and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[17] Zhang, H., Zhang, D., Guan, J., Wang, D., Tang, M., Ma, Y., and Xia, H. A flexible wearable strain sensor for human-motion detection and a human–machine interface. *Journal of Materials Chemistry C 10*, 41 (2022), 15554–15564.