Identifying Phone Activity Through Magnetometer

MIHAEL TRAJBARIČ and MILENKO OBRADOVIĆ, University of Ljubljana, Faculty of Computer and Information science, Slovenia

Magnetometer readings are not considered sensitive, hence no user permission is required to access the readings. We showed its readings could also be exploited to measure magnetic emissions, produced by the device's internal components. These readings can then be exploited to predict if internal communication modules (WiFi, Bluetooth and cellular antenna) are under heavy load or idle and therefore give the application the ability to infer the state of communication without actually gaining permissions. We leveraged support vector machine and Random forest to achieve over 95% accuracy for a single device on a known location and 90% accuracy on a new location. However, leveraging models to an unknown device remains challenging.

Additional Key Words and Phrases: mobile sensing, magnetometer classification, activity classification

ACM Reference Format:
Mihael Trajbarič and Milenko Obradović. 2023. Identifying Phone Activity Through Magnetometer. 1, 1 (January 2023), 6 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

User privacy is an important feature of modern smartphones. Applications cannot easily access readings from different sensors or data about the hardware of the phone. Access must be granted by the user via permissions.

However, not all sensors have equal protection, since not all data is considered equally private. One of the least protected sensors is magnetometer[1], which samples 3-dimensional ambient magnetic field, together with estimated hard-iron biases for calibration. The primary use of a Magnetometer is to serve as a compass and help with navigation, but it can also sense the presence of electric wiring or other outside magnetic emissions. None of this is considered private data.

Since a smartphone is also an electric device, it should also produce magnetic emissions, which are a direct result of electric processes inside electric components. Different loads on components should produce different magnetic emissions. Is it possible, to leverage magnetometer readings to obtain information about activity on the phone?

In the scope of this research, we focused on communication modules (WIFI, Bluetooth, and cellular data). Based on the readings from the magnetometer, we are trying to predict if all 3 of these modules are **idle** (turned off) of **under heavy load**.

[1]https://source.android.com/docs/core/interaction/sensors/sensor-types#magnetic_field_uncalibrated

Authors' address: Mihael Trajbarič, mt3714@student.uni-lj.si; Milenko Obradović, mo5325@student.uni-lj.si, University of Ljubljana, Faculty of Computer and Information science, Večna pot 113, Ljubljana, Slovenia, 1000.

2 RELATED WORK

The exploitation of magnetometer sensors on android devices has been actively researched in the past. [Kuk et al. 2018] leveraged magnetometer emissions from nearby phones for human contact detection. Magnetometer proved to be ideal for contact detection since it can only detect emissions from phones, that are near enough to be considered contacts for transmission of disease. Similar approaches have also been investigated for general localization [McFee et al. 1994].

Similarly, [Perez et al. 2019] has leveraged magnetometer readings to identify mobile devices. it has been proven magnetometer fingerprint, obtained with a nearby phone or from the sensor on the same device, is unique for every device. Identity has been inferred with simple machine learning models (Random Forests, Support Vector Machines) on only a couple of data points.

Leveraging magnetometer readings for activity prediction is widely researched. Due to aggregate CPU usage and power consumption traces being application- and website-specific, they can be used to predict applications and websites on smartphones. While traces cannot be directly observed since they are protected and need permissions, the electromagnetic activity causes significant disturbances in magnetometer measurements and can be exploited[Matyunin et al. 2019].

Application activity can even be predicted from another device. This effect is greater on a laptop, where power consumption is greater which leads to greater electromagnetic activity. Although challenging due to noise, certain events, for instance, application launching or sustained operations can be predicted from a nearby smartphone using only magnetometer readings [Ji et al. 2023].

While other approaches focus on CPU-related power consumption, we focus on magnetic disturbances, caused by communication modules. We focus on Wifi, Bluetooth, and cellular data modules. Our main contributions are:

- A novel exploitation of magnetometer readings for identifying communication state. We achieve over 90% accuracy on a single device and unknown location.
- A data-collection app, which guides users through different data-gathering phases.
- A dataset with 96 samples, collected with 4 different phones, 14 locations, and with multiple positions on every location.

3 DATA COLLECTION

Application code for the Data Collection Application is available at our GitHub repository[2].

Prior to gathering data and creating a data set, used in the machine learning process to create models, we needed to write an application for android that can be installed on the device and collect data about magnetometer sensor readings on the device.

[2]https://github.com/magnetometer-identification/MagnetometerDataCollection

The data structure has been envisioned as a JSON file that contains a dictionary of lists of records on the x, y, and z axes, timestamp, device_id, and some data that was found to not be useful for this paper, like accuracy and bias values on axes.

Originally the plan was to record magnetometer readings in 6 different stages of phone states, which was later downgraded to only 2 stages. Stage 5 (All-Off) presents an idle state, where mobile data, WiFi, and Bluetooth are off and Airplane mode is on. Stage 6 (All-On) is a stage with a full load, where all communication modules are turned on and Airplane mode is off. The stage labels (5 and 6) remain from the initially conceptualized 6-stage process. Return to 6 stages is possible with a few modifications in code.

Our application is connected to a Firebase server that stores files with recordings. Files can be organized into subfolders, according to the user's preferences from the app.

The data gathering time window length is set to one minute, but can also be easily changed with the modification of a few lines of code.

Before using the application it is recommended to make sure that a phone battery is above 20%.

3.1 Design of application

The data-gathering application design is basic and simple to use. It consists of two activities and doesn't have background capabilities, as the data-gathering time is just one minute.

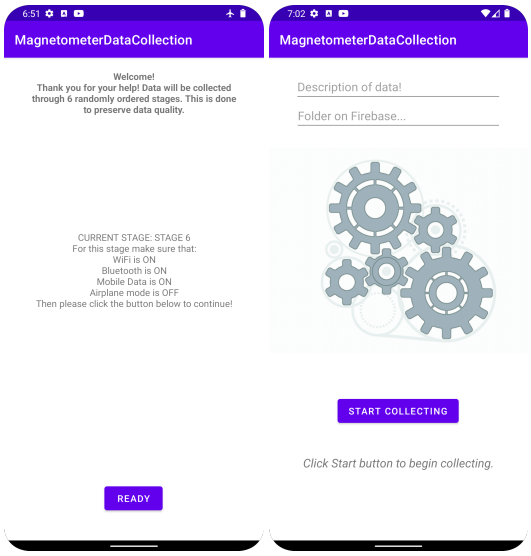

Fig. 1. Activities 1 and 2.: The home screen is on the left and the Configuration screen is on the right.

The first activity (Figure 1 on the left) that is visible upon opening the app is the stage-configuration activity. There are brief instructions on how to put the phone in the appropriate stage. After the conditions are fulfilled, the user can click on the button *Ready* and continue to the second activity Stages (in this case stages 5 or 6) are served in random order, to minimize the effect of the order on data.

Due to changes in how permissions work among different versions of android, we chose not to automatically turn on or off the necessary communication modules and left this setting to the user.

After passing to the second activity (Figure 1 on the right), a user needs to write a brief description of the location, device, and name of the folder on Firebase. If the folder doesn't exist it will be created.

Data will be collected after the user clicks the start button. In case stage 5 is recorded as second, the user will be warned when to turn on WiFi or mobile data so the files can be sent to the server after the collection of data from both stages.

3.2 Development of application

For the development, we used Android Studio and the language Kotlin. When we started, the main idea was to use the AWARE library to extract magnetometer readings. But that turned out to be inefficient as AWARE doesn't offer uncalibrated data or bias corrections which we needed. Later we proceeded to use of stock Magnetic field uncalibrated sensors, accessible directly without any middleware.

Over the lifecycle of the application, we encountered a few bugs due to human error. One of those mistakes the second stage always contained twice the amount of data. The reason for the bug was failed re-initialization of a list with sensor readings, which was implemented when migrating from AWARE to stock android sensors.

Another problem we encountered and could not fully mitigate was the power-saving mode on some devices. When the app is collecting data, a user usually leaves the app in the background and opens another app (Youtube) to put WIfi under load. Some phones have built-in battery savers which is turning off background apps. This produced weird behavior and restarting activities. We overcome the problem by running data-collection on full battery.

3.3 Data gathering process

To make sure that data is collected in optimal conditions and labeled correctly, we agreed on the procedure of gathering.

The phone should be placed on a steady surface, not necessarily flat, but it is important that the phone is not moved or touched a lot. So the data is not corrupted. It is a good practice not to move electronic devices that are near devices with running apps.

Data were collected inside and outside of the living and working space. Through experimentation, we realized that it would be useful to rotate the phone on multiple takes from the same location to make the position of the phone less relevant.

In an ideal situation, the phone should be rotated 45 degrees on each take, and there would be 8 takes. Although 3 or 4 rotations should be enough. When the application is collecting data on the phone stage 6 (All-On), the phone was connected to Bluetooth earbuds before collecting started and after that, we created some internet (WiFi) traffic by playing video online.

4 DATA ANALYSIS

4.1 Dataset

The dataset consists of 96 test cases, equally distributed across both classes. It was collected with 4 mobile devices across 15 different, as depicted in Table 1. Data were recorded in a mix of public and

Table 1. Test cases per location, grouped by device. Data was gathered in sessions, consisting of 2 test cases per session, one from each class, hence the number of cases is always even. Multiple sessions were recorded on most of the locations. 'Rotations' depict the test device was rotated for some angle around Z-axis between subsequent sessions.

Device	Location	Rotations	# of test-cases
GooglePixel6	DormRoom7	True	16
GooglePixel6	DormRoom2	True	14
GooglePixel6	DormRoom4	False	4
GooglePixel6	Kitchen3	False	4
GooglePixel6	Park	False	4
RedmiNote8PRO	Kitchen1	True	6
RedmiNote8PRO	LivingRoom2	True	6
RedmiNote8PRO	Bathroom	False	2
RedmiNote8PRO	LivingRoom1	False	2
SamsungGalaxyA51	DormRoom1	False	6
SamsungGalaxyA51	DormRoom5	True	6
SamsungGalaxyA51	DormRoom6	False	6
SamsungGalaxyA51	Restaurant	False	6
SamsungGalaxyS6	Kitchen2	True	8
SamsungGalaxyS6	DormRoom3	True	6

private places, outdoors and indoors with multiple sessions per location.

To increase model robustness[Chang et al. 2020], we collected multiple sessions with different rotations in some locations.

Every test case contains 1-min-long recording of magnetometer readings. The number of samples per recording depend on max sample rate and varies across test devices in (4000, 8000) range.

Every recording consist of 3 signals of *uncalibrated* sensor readings along XYZ-axes and 3 signals of estimated *hard-iron biases* along the same XYZ-axes.[3]. This gives us 6 raw signals, *X_UnCal*, *Y_UnCal*, *Z_UnCal*, *X_Bias*, *Y_Bias* and *Z_Bias*,

4.2 Data-processing pipeline

All the data and code for data-processing is available at our github repository[4]. Raw data in JSON format is manually downloaded from Google Firebase portal[5] and transformed to CSV. Every uncalibrated signal (X_UnCal, Y_UnCal and Z_UnCal) is normalized. We experimented with both *min-max normalization* (Equation 1) and *standardization* (Equation 2).

$$x' = \frac{x - min(x)}{max(x) - min(x)} \tag{1}$$

$$x' = \frac{x - \mu}{\sigma} \tag{2}$$

Each test-case (length 60 seconds) is sub-sampled with rolling window (width = 10s, stride = 0.5s).

[3]https://source.android.com/docs/core/interaction/sensors/sensor-types#magnetic_field_uncalibrated
[4]https://github.com/magnetometer-identification/DataAnalysis
[5]https://console.firebase.google.com/

4.3 Feature generation

We introduce rotation-invariant feature **intensity**, defined as L2 norm of *X_UnCal*, *Y_UnCal* and *Z_UnCal* (Equation 3).

$$\|X\|_2 = \sqrt{x_1^2 + x_2^2 + x_3^2} \tag{3}$$

We transform all 3 signals (X, Y, Z) and intensity from time domain to frequency domain using Fast Fourier Transform (FFT)[6].

We engineered 2 distinct feature-sets. **spectral feature-set** contains first 50 frequencies from intensity signal, obtained with FFT. **statistical feature-set** consist of descriptive statistics (mean, std, min, max, 25%, 50 % and 75%), calculated on both time- and frequency-domain representations of dataset. In time domain, due to static nature of bias signals, we only calculate statistic on 3 un-calibrated signals and intensity.

4.4 Feature selection

We evaluated every feature using correlation between the feature and target class. We order features by collation and chose first k. Choice of k depends on the choice of training feature-set and was chosen optimally for every experiment.

4.5 Classifiers

We used **Random Forrest Classifier** with max_depth = 3 and **Support Vector Machine** with one-vs-one decision function. For both classification techniques, we used implementation from scikit-learn library[7].

5 EXPERIMENTS & RESULTS

5.1 Single device on known location

For the first experiment we chose a single device. A single session with two cases (one all_on and one all_off) was chosen for test and all the remaining sessions from all locations for this device were used for training. We experimented on standardized test set and on both spectral and statistical feature-sets. Results are in Table 2. The best results were obtained on a standardized spectral feature-set, where several test devices were able to achieve accuracy and F1 scores over 90% with first 10 features. RedmiNote8PRO even achieved accuracy and F1 score over 97% with only first feature on both SVM and RF classifier.

5.2 Single device on a new location

For the experiment with single device on a new location we took all samples from one location as test and all samples for one device from other locations for train. We averaged results over all the location of single device. Results are in Table 3.

Redmi retained the same accuracy of 1.0, while Pixel6 and GalaxyA51 experienced drop of 5% in accuracy in comparison to results from known location. Interestingly, results from GalaxyS6 increased.

5.3 Multiple devices on new location

We chose all sessions from one location of one device as a test set and all the other sessions from all devices as a train. We trained

[6]https://docs.scipy.org/doc/scipy/tutorial/fft.html
[7]https://scikit-learn.org/stable/

Table 2. Accuracy and F1 scores for classification on known location. Number beside name of device depict number of spectral features used for training. One session was chosen for test and remaining sessions from all locations for one device were used as for training.

Device_features	SVM_acc	SVM_f1	RF_acc	RF_f1
Redmi_1	0.990	0.989	0.977	0.972
Pixel6_1	0.929	0.898	0.926	0.890
GalaxyA51_1	0.739	0.700	0.756	0.690
GalaxyS6_1	0.842	0.827	0.861	0.876
Redmi_10	1.0	1.0	1.0	1.0
Pixel6_10	0.957	0.937	0.950	0.931
GalaxyA51_10	0.941	0.838	0.956	0.871
GalaxyS6_10	0.888	0.890	0.891	0.890

Table 3. Accuracy and F1 scores for classification on a new location. All test cases from one location were chosen for test and remaining sessions from all other locations for one device were used for training. First 10 features from spectral features set were used for training and classification. Results were averaged over all locations of a single device.

Device	SVM_acc	SVM_f1	RF_acc	RF_f1
Redmi	1.0	1.0	1.0	1.0
Pixel6	0.907	0.873	0.894	0.874
GalaxyA51	0.899	0.887	0.816	0.807
GalaxyS6	0.933	0.927	0.917	0.911

on both standardized spectral (Table 4) and statistical feature sets (Table 5).

Results vary across different devices. The best results with spectral features set achieved Pixel6, while the worst results are from Redmi. Interestingly, the best results on the statistical feature set were achieved on Redmi, followed by Pixel6, GalaxyA51 and GalaxyS6.

Overall accuracy, averaged over all the locations is higher on spectral feature-set (SVM_acc = 74%, RF_acc = 71%) compared to statistical feature-set (SVM_acc = 71%, RF_acc = 69%).

5.4 New device

Classification on a new device included testing on all cases from one device and training on all cases from other devices. We trained on spectral feature-set. Results are in Table 6.

None of the models were successful to classify test cases on Redmi, both Random Forest and Support Vector Machine achieved F1 score of 0. Generally, RF performed better them SVM with 75% Accuracy and 73% F1 score averaged over other 3 devices. With average accuracy of 56% and F1 score 65%, averaged over other 3 devices, SVM did not performed significantly better then lucky guess.

5.5 Effect of normalization and standardization

We introduced normalization to overcome variability in locations and hardware properties among different sensor. We experimented with two different normalization techniques, min-max normalization and standardization. Experiments were done on classification of

Table 4. Accuracy for classification on first 10 features from a spectral feature-set on a new location, trained on all devices. All test cases from one location were chosen for test and remaining sessions from all other locations for all devices were used for training.

Device	Location	SVM	RF
GooglePixel6	DormRoom2	0.929	0.943
GooglePixel6	DormRoom4	0.998	0.989
GooglePixel6	DormRoom7	0.839	0.708
GooglePixel6	Kitchen3	1.000	1.000
GooglePixel6	Park	0.998	0.966
RedmiNote8PRO	Bathroom	0.695	0.500
RedmiNote8PRO	Kitchen1	0.531	0.501
RedmiNote8PRO	LivingRoom1	0.520	0.500
RedmiNote8PRO	LivingRoom2	0.515	0.502
SamsungGalaxyA51	DormRoom1	0.602	0.603
SamsungGalaxyA51	DormRoom5	0.819	0.832
SamsungGalaxyA51	DormRoom6	0.569	0.660
SamsungGalaxyA51	Restaurant	0.701	0.651
SamsungGalaxyS6	DormRoom3	0.574	0.603
SamsungGalaxyS6	Kitchen2	0.876	0.879

Table 5. Accuracy for classification on first 10 features from a statistical feature-set on a new location, trained on all devices. All test cases from one location were chosen for test and remaining sessions from all other locations for all devices were used for training.

Device	Location	SVM	RF
GooglePixel6	DormRoom2	0.816	0.835
GooglePixel6	DormRoom4	0.786	0.929
GooglePixel6	DormRoom7	0.875	0.931
GooglePixel6	Kitchen3	0.857	0.952
GooglePixel6	Park	0.952	0.952
RedmiNote8PRO	Bathroom	1.000	0.500
RedmiNote8PRO	Kitchen1	0.800	0.560
RedmiNote8PRO	LivingRoom1	0.750	0.500
RedmiNote8PRO	LivingRoom2	0.808	0.615
SamsungGalaxyA51	DormRoom1	0.667	0.893
SamsungGalaxyA51	DormRoom5	0.643	0.548
SamsungGalaxyA51	DormRoom6	0.443	0.757
SamsungGalaxyA51	Restaurant	0.488	0.702
SamsungGalaxyS6	DormRoom3	0.467	0.367
SamsungGalaxyS6	Kitchen2	0.370	0.370

Table 6. Accuracy and F1 scores for classification on first 10 features from a spectral feature-set on a new device, trained on all devices. All test cases from one device were chosen for test and remaining sessions from all other devices were used for training. Results were averaged over devices.

Device	SVM_acc	SVM_f1	RF_acc	RF_f1
GooglePixel6	0.592	0.713	0.795	0.830
RedmiNote8PRO	0.501	0.000	0.501	0.000
SamsungGalaxyA51	0.627	0.595	0.713	0.583
SamsungGalaxyS6	0.478	0.622	0.750	0.770

Table 7. Effect of normalization tehnique on classification accuracy.Training was perfomed on first 10 features from a spectral feature-set on a new device. All test cases from one device were chosen for test and remaining sessions from all other devices were used for training. Results were averaged over devices.

Device_method	raw	min-max	standardized
GooglePixel6_SVM	0.529	0.501	0.592
RedmiNote8PRO_SVM	0.812	0.248	0.501
SamsungGalaxyA51_SVM	0.556	0.552	0.627
SamsungGalaxyS6_SVM	0.534	0.510	0.478
GooglePixel6_RF	0.529	0.501	0.592
RedmiNote8PRO_RF	0.812	0.248	0.501
SamsungGalaxyA51_RF	0.556	0.552	0.627
SamsungGalaxyS6_RF	0.534	0.510	0.478

data from new device, trained on entire spectral feature-sets. Results are in Table 7.

Surprisingly normalization did not improve accuracy when classifying with SVM, which achieved average accuracy of 61% on raw data compared to 55 % on standardized data. Accuracy did improve on RF from 63% on raw data to 69 % on standardized data. Min-max normalization did not improve results in any of experiments.

5.6 Effect of number of rotations

We chose samples from two locations from Pixel6 (DormRoom2 and DormRoom7), for which we recorded sessions with 7 and 8 different rotations respectively. For every number of rotations, we trained on samples from 100 randomly chosen rotations from location DormRoom7 and tested on all cases from DormRoom2. We trained SVM and RF with first 10 spectral features for every experiment.

Results (Figure 2) show number of rotations does not effect performance of Random Forest classifier, while it has significant effect on performance of SVM.

5.7 Effect of number of features on accuracy

Results in Figure 3 show even first feature achieves respectable accuracy, while next few features improve accuracy for 2%. Similar results happen on statistical dataset.

For this experiment, we repeated Classification on new location experiment for every number of features from (1, 15). Features were ordered using correlation and first i features were selected. Experiment was performed for both spectral and statistical feature-set both on standardized dataset from Pixel6.

6 CONCLUSIONS & FUTURE WORK

Our experimentation showed both Support Vector Machine and Random Forrest classifier can predict correct state of communication-modules on known device. This works both on known location as well as on a new location.

We showed outside interference can be mitigated using robust data recording (multiple rotations), post processing (signal standardization) and employing features from statistical and spectral domain.

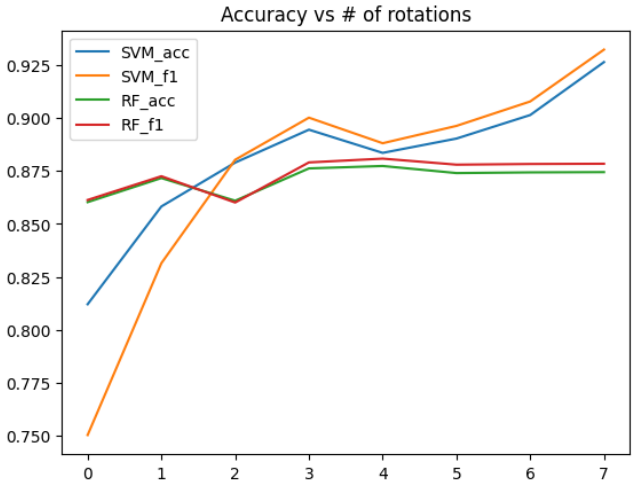

Fig. 2. Accuracy vs number of rotations. Trained on random subsample from list of rotations from location DormRoom7, averaged over all subsamples of same size. Tested on all samples from location DormRoom2. Trained with both SVM and RF on first 10 features from spectral feature-set.

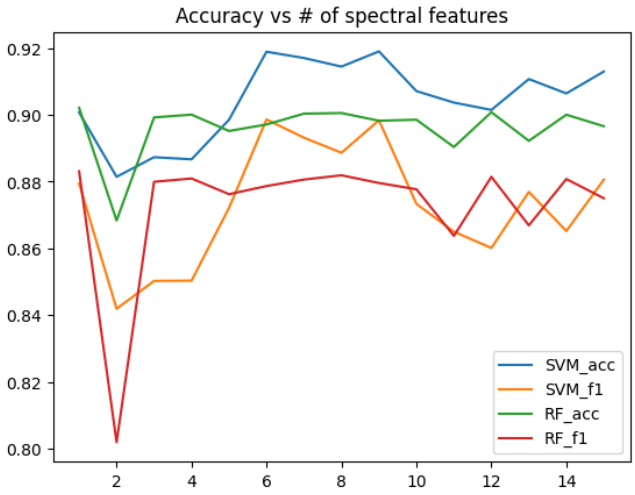

Fig. 3. Accuracy vs number of features for Classification on new location. Classified with both SVM and RF. Trained with a first k features, according to correlation. Trained on spectral feature-set from Pixel6 dataset.

The most challenging part remains classification on a new device. We observed major difference among test devices, which could be contributed to different hardware components (magnetometer, communication modules).

Another issue was not completely balanced dataset. While classes were balanced over devices and locations, number of locations or cases per device was not. We do not believe this had a huge impact on results overall. Among 3 devices with lower number of test cases, only RedmiNote8PRO did not signifcantly improve on random guess

(RF_acc = 0.501) while performance on GalaxyS6 and GalaxyA51 were not far behind Pixel6 (RF_acc of 0.75 and 0.71 vs 0.795 on Pixel).

Future work should improve on dataset quality and size. A completely balanced dataset should contain equal number of cases per location, with multiple rotations per location. Another emprovement could be collecting data on all devices on the same set of locations.

Another possible improvement would be to employ deep learning models on images, obtained from spectral dataset.

7 AUTHOR CONTRIBUTIONS

Milenko designed and implemented the application for data collection. He also collected most of the data. Mihael provided feedback on the application, designed and implemented data pipeline, did data analysis and experimentation. He also collected additional 10 data samples.

REFERENCES

Youngjae Chang, Akhil Mathur, Anton Isopoussu, Junehwa Song, and Fahim Kawsar. 2020. A Systematic Study of Unsupervised Domain Adaptation for Robust Human-Activity Recognition. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 4, 1, Article 39 (mar 2020), 30 pages. https://doi.org/10.1145/3380985

Xiaoyu Ji, Yushi Cheng, Wenyuan Xu, Yuehan Chi, Hao Pan, Zhuangdi Zhu, Chuang-Wen You, Yi-Chao Chen, and Lili Qiu. 2023. No Seeing is Also Believing: Electromagnetic-Emission-Based Application Guessing Attacks via Smartphones. *IEEE Transactions on Mobile Computing* 22, 2 (2023), 1095–1109. https://doi.org/10.1109/TMC.2021.3092209

Seungho Kuk, Junha Kim, Yongtae Park, and Hyogon Kim. 2018. Empirical Determination of Efficient Sensing Frequencies for Magnetometer-Based Continuous Human Contact Monitoring. *Sensors* 18, 5 (2018). https://doi.org/10.3390/s18051358

Nikolay Matyunin, Yujue Wang, Tolga Arul, Kristian Kullmann, Jakub Szefer, and Stefan Katzenbeisser. 2019. MagneticSpy: Exploiting Magnetometer in Mobile Devices for Website and Application Fingerprinting. In *Proceedings of the 18th ACM Workshop on Privacy in the Electronic Society* (London, United Kingdom) *(WPES'19)*. Association for Computing Machinery, New York, NY, USA, 135–149. https://doi.org/10.1145/3338498.3358650

J.E. McFee, R.O. Ellingson, and Y. Das. 1994. A total-field magnetometer system for location and identification of compact ferrous objects. *IEEE Transactions on Instrumentation and Measurement* 43, 4 (1994), 613–619. https://doi.org/10.1109/19.310176

Beatrice Perez, Mirco Musolesi, and Gianluca Stringhini. 2019. Fatal Attraction: Identifying Mobile Devices through Electromagnetic Emissions. In *Proceedings of the 12th Conference on Security and Privacy in Wireless and Mobile Networks* (Miami, Florida) *(WiSec '19)*. Association for Computing Machinery, New York, NY, USA, 163–173. https://doi.org/10.1145/3317549.3319726