

An accelerometer-based privacy attack on smartphones

Roberto De Prisco¹, Alfredo De Santis¹ and Rocco Zaccagnino¹

¹University of Salerno, Computer Science Department, Via Giovanni Paolo II, 132 - 84084 Fisciano (SA), Italy

Abstract

Most smartphones are equipped with an *accelerometer sensor*. There are numerous scenarios in which this sensor can be very useful. However it can also represent a privacy threat. Indeed, the measurement of the device vibrations can be exploited to detect private information. The attack can be favored by the fact that this specific sensor is normally not considered a “dangerous” one and also by the fact that the measurements of today’s sensors are quite accurate.

Recently many research studies have focused on the task of inferring information from the accelerometer measurements. There are several settings that can be considered and several final goals; in this paper we consider the specific case of recognizing words that the device itself is reproducing through its loudspeakers. A recent paper has considered this scenario and has proposed a recognizer, based on Convolutional Neural Networks, for single digits, single letters and a small set of “hot words”.

Following such a research direction, in this paper, we provide an improved recognizer for single letters and digits. We performed an evaluation study to assess the effectiveness of the proposed attack. Results show that the system outperforms the previous approach. We also propose a generalization whose goal is that of recognizing entire words, or even sentences, not by means of a dictionary, but by first recognizing syllables and then locate sequences of syllables that correspond to words. We provide preliminary results in this direction.

Keywords

Mobile security, Speech privacy attack, Deep learning

1. Introduction

The idea of *smartphone*, i.e., a device integrating both telephony and some computer capabilities, dates back to 1993, when IBM designed the first smartphone ever: *Simon*¹. Since 1993, smartphones have increasingly become an essential component of our daily life to the point of assuming the role of interface with the rest of the world in several situations, thanks to the many communication possibilities made available. Among these, voice communication is clearly the main one. Because of this, operating systems usually restrict the access to the microphone by placing its usage at the highest permission level². The search for security vulnerabilities associated with smartphones has moved over time towards other types of sensors,

Itasec21: Italian conference on Cybersecurity, April 07–09, 2021, Online

✉ robdep@unisa.it (R. D. Prisco); ads@unisa.it (A. D. Santis); rzaccagnino@unisa.it (R. Zaccagnino)

🌐 <https://docenti.unisa.it/003550/home> (R. D. Prisco); <https://docenti.unisa.it/000769/home> (A. D. Santis);

<https://docenti.unisa.it/023039/home> (R. Zaccagnino)

🆔 0000-0003-0559-6897 (R. D. Prisco); 0000-0001-8962-1919 (A. D. Santis); 0000-0002-9089-5957 (R. Zaccagnino)

© 2021 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings (CEUR-WS.org)

¹<https://www.businessinsider.com/worlds-first-smartphone-simon-launched-before-iphone-2015-6>

²<https://developer.android.com/guide/topics/sensors/sensorsoverview>.

and in particular towards *motion sensors*. This is probably due to the fact that such sensors are normally considered not dangerous and thus their access is generally unrestricted (for example in Android, at least up to the current version, there is no need to ask a permission to access the accelerometer).

The technological evolution of motion sensors integrated in smartphones has resulted in the development of many smart applications which infer information from them to solve tasks. For example, human activity recognition [1, 2], age group detection [3], health monitoring and diagnosis [4], gender recognition [5], and continuous authentication with privacy preservation [6].

It is easy to find in recent literature many research papers that focus on inferring private information from “innocuous” sensors, among which the accelerometer is one of the most studied. Several research studies exploit motion sensors for eavesdropping on keystrokes, touch input and speech, without the requirement of system permissions, and without hacking into the operating system for gaining access to the administrator authority [7, 8, 9, 10, 11, 12].

Most of the work that focus on speech recognition considers the scenario in which the speech to be recognized comes from external speakers. Although this is certainly an interesting case, it is quite different from the case in which the speech to be recognized comes from the measuring device itself. In this paper we focus on this latter case: the device that measures the accelerometer signals is the same device that produces the speech to be recognized. Although this specific case doesn’t capture the case in which a user is speaking through the device, it captures many other situations, in particular the reproduction of vocal messages on the device. In order to have accelerometer measurements that depend only on the sounds being reproduced it is necessary that the device not be held in the hand of the user. We will assume that the device is placed by itself on a table; this scenario is referred to as the *table setting*. The table setting scenario doesn’t seem to have been considered, apart from [9]. In that paper the authors focus on the recognition of single letters and digits and on a small set of *hotwords*, leaving almost downright unexplored the problem of recognizing entire words in real conversations.

In this paper we make a small step in this direction. We provide an accelerometer-based recognizer that works in the table setting scenario. For the specific problem considered, we designed a recognizer for single letters and digits based on a more compact deep learning model with respect to the one provided in [9], and we make a step towards the recognition of entire, arbitrary words. More specifically, the main contributions of the paper are the following:

- a novel *deep learning*-based system that, using a simple, custom built, CNN and starting from the spectrogram representation of acceleration signals, learns to recognize the *speech units*, i.e., those basic components of the language which combined produce words (letters, digits and syllables).
- a proposal of a *general method* to recognize words in speech conversations; specifically we define a set of basic syllables (speech-unit) and build a recognizer with the same technique used for the single letters and digits; then we try to recognize specific sequences of syllables that make up the words. We also provide a simple implementation of such a method. More clever and effective implementations are left as future work.

The tests have been conducted on an Android smartphone. We have written an Android app to collect the accelerometer readings. Although we have used Android as test system,

the overall method described is general and does not depend on the specific operating system (except for the access to the accelerometer which, in Android, is allowed without requesting any permission). Results on different systems can vary as a function of the hardware characteristics of the devices.

The rest of the paper is organized as follows. In Section 2.1, we describe some relevant works in the field of speech privacy attack in Android using motion sensors. In Section 3, we describe the methodology followed to define the leaning-based speech units recognizer for single letters and digits. In Section 4, we discuss a generalization to words. Finally, in Section 6 we provide concluding remarks with some future directions.

2. Related work and threat model

2.1. Related work

Several speech privacy attacks using motion sensors have been proposed. The discriminating elements between the various studies are the type of motion sensors exploited, and the setup in which such the sensor is stimulated to collect information regarding the speech signals. An extensive study of the accelerometers and gyroscopes response to speech signals in various setups is proposed in [8]. The authors stimulate both sensors with *human-rendered*, *laptop-rendered* and (*external*) *loudspeaker-rendered* speech signals traveling through the air or a solid surface. Results show that only loudspeaker-rendered speech signals traveling through a solid surface can create noticeable impacts on motion sensors.

In [11] the authors proposed a study in which (*i*) the smartphone is placed on the same solid surface as the external loudspeaker used to reproduce the audio, (*ii*) the smartphone's gyroscope is used to collect the surface vibrations caused by the speech signals emitted by the loudspeaker, and (*iii*) the captured information were used to conduct speech recognition and speaker identification. Due to the low sensitivity to the gyroscope with respect to the surface vibrations, and to its limited sampling rate (200Hz), the performance for the recognition task does not achieve high success rates (65% for the speaker dependent case and up to 26% for the speaker independent case).

In [12] the authors proposed a setup in which (*i*) the user speaks to a smartphone held in her hand or placed on a desk, (*ii*) the accelerometer is used to collect speech signals traveling through the air, and (*iii*) the accelerometer readings are used to conduct *hot words detection* ("Okay Google" and "Hi Galaxy"). However, results show that the accelerometer may not be able to collect sufficient information through airborne vibrations, suggesting that the speech signals traveling through the air are unlikely to have any noticeable impact on motion sensors.

In the above cited work the device emitting the sound (external speaker, smartphone, computer) is different from the one (smartphone) that captures the accelerometer readings. An interesting case is the one in which the two devices are the same; that is the smartphone is used both to reproduce the speech signal and to record the accelerometer readings. This setting is referred to as the *table setting* imagining that the device is placed by itself on a table while performing the experiments. The table setting has been considered by [9], where the readings coming from the accelerometer are analyzed using deep learning techniques. In [9] the authors investigate a comprehensive set of factors and address them with effective preprocessing

approaches. The specific technique used in [9] transforms the accelerometer readings into images and then uses standard approaches from computer vision to recognize the images; more specifically a recognizer based on *DenseNet* is used. The recognizer built in [9] focus on the recognition of single letters and single digits and on words coming from a small set (namely "password", "username", "social", "security", "number", "email", "credit" and "card").

2.2. Setting

We follow the case considered in [9], i.e., we consider the table setting in which the targeted smartphone is used both to reproduce the speech signal and to record the accelerometer signals.

Threat model. We assume that the *victim's* smartphone contains our SpyApp that exploits the accelerometer to record its measurement during the reproduction of a speech with the smartphone placed on a table. This can be the case in office or home environments, where conversations are often based on the exchange of voice messages. Thus the captured speeches include voice messages from the contacts of the victim, since the spy app on the smartphone will record data coming from any source, such as *voice memo* listened by the user, *location information* emitted by the smartphone speaker during support voice guidance, and *music/video preferences* which can be analyzed with the goal of constructing the user's listening and watching habits.

We are not concerned about how the SpyApp can be installed on the smartphone and also about the fact that the victim must play the speech signal placing the smartphone on a surface. The goal of the research is that of understanding if in such a situation it is possible to infer the words from the captured measurements of the accelerometer. We first consider the basic case in which the *speech unit* that we want to recognize are only single letters and single digit. As a second step we want to generalize the recognition to arbitrary words.

The SpyApp continuously collects the accelerometer measurements and sends the data to a server. On the server we implement the recognizer that we will describe in later sections.

Figure 1 is a graphical representation of the system considered.

Implementation. For the actual implementation we used an Android smartphone requesting the operating system to use the fastest sampling rate (`SENSOR_DELAY_FASTEST`). Thus the actual sampling rate is determined by the hardware. For the specific device that we used, a Samsung S8 2017 Quad-core 2.3 GHz + Quad-core 1.7 GHz, chipset 8895 Samsung Exynos, 64bit, we have a sampling rate of 420 Hz, which is more than enough to sample the human voice.

3. A deep learning-based speech-units recognizer

The speech-units recognizer has to solve the following recognition or *classification* problem: given the accelerometer measurements of a speech unit reproduced by the smartphone, recognize the speech unit. The recognizer we propose is based on the same methodology described in [9]; however we experimented with several possible variants and identified a solution that works better (at least in the test we conducted) than the one described in [9]. The idea at the base of

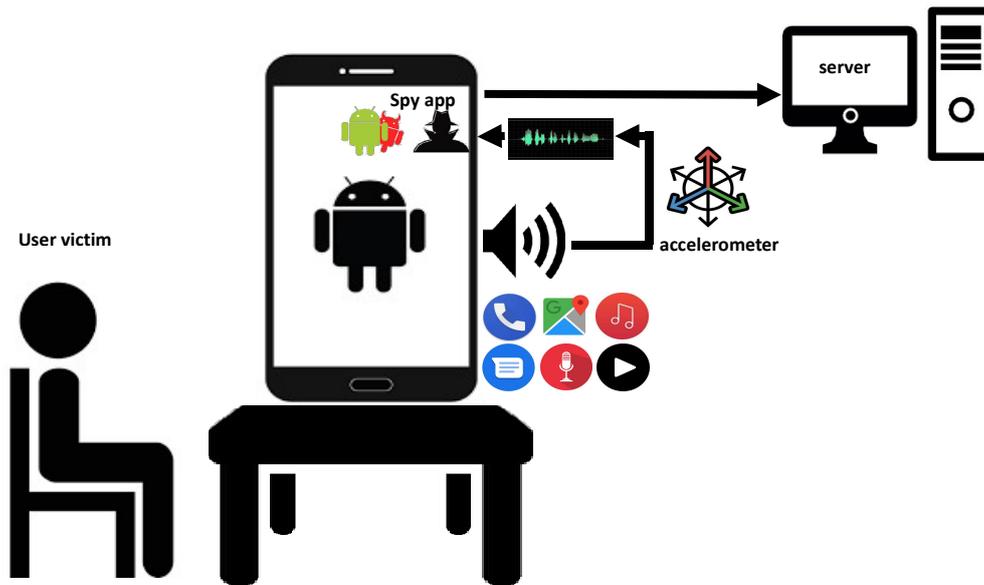


Figure 1: Threat model of the proposed side channel attack

the methodology described in [9] is that of representing the accelerometer signals as images and exploit the powerful deep learning models studied in computer vision in order to solve the classification problem for the speech units. Among the deep learning models commonly used in computer vision we have *VGG* [13], *ResNet* [14], *Wide-ResNet* [15], *DenseNet* [16]. All of these are CNNs with different characteristics. The one used in [9] is the *DenseNet*, but the exact type it is not specified. All of the above CNNs are quite powerful but also computationally expensive since they are made up of a considerable number of layers. For example *DenseNets* can have 121, 169 or more layers. We propose to use a CNN with a considerable smaller number of layers, namely 12. Thus the CNN we propose to use for the speech-units recognizer is not a standard one. We call the proposed model *AccCNN*. In order to evaluate *AccCNN*, we have implemented other speech-units recognizers based on the above cited CNNs: *VGG*, *ResNet*, *Wide-ResNet* and *DenseNet*. The last one is thus an implementation of the approach described in [9]; we remark that we used a *DenseNet* with 121 layers - in [9] the number of layers used is not specified. The tests that we have conducted show that, despite being much more simple with respect to the others, *AccCNN* exhibits better performances.

In this section we provide details about the construction of the network and about the tests.

3.1. The CNNs

We have experimented with several alternative CNNs: some standard CNNs and a custom designed CNN, which we describe in this section. In later sections we report the results of the experiments.

3.1.1. Standard CNNs

Convolutional Neural Networks are a multi-layer neural network designed to recognize visual patterns directly from pixel images with minimal preprocessing. In recent years, we have witnessed the birth of numerous CNNs. Among those, we find VGG, ResNet, Wide-ResNet and DenseNet. These networks have gotten so deep that it has become extremely difficult to visualize the entire model. Since they are standard and there are public libraries that implement them, we use them as black boxes. There exists two types of VGG, namely VGG16 and VGG19. We have considered VGG19, that have 16 convolution layers and 3 dense layers, for a total of 19 layers. ResNet were introduced to answer the following question: “why by adding more layers to deep neural networks does the accuracy not improve, but it actually gets worse?”. Intuitively, deeper neural networks should not perform worse than shallow ones, or at least not during training when there is no risk of overfitting. However, as the depth of the network grows this is not always true. Thanks to the innovation introduced by ResNet, we can now build networks of countless layers. Several variants have been proposed in literature. In this work we have considered ResNet50 consisting in 50 layers. ResNets were shown to be able to scale up to thousands of layers and still have improving performance. However, each fraction of a percent of improved accuracy costs nearly doubling the number of layers, and so training very deep residual networks has a problem of diminishing feature reuse, which makes these networks very slow to train. To tackle these problems, WideResNet have been introduced. We have used a WideResNet consisting in 40 layers. In DenseNet, each layer obtains additional inputs from all preceding layers and passes on its own feature-maps to all subsequent layers. Concatenation is used. The idea is that each layer is receiving a “collective knowledge” from all preceding layers. We have considered a well-known DenseNet, named Dense121, consisting in 121 layers.

3.1.2. AccCNN: a custom CNN

Running several tests and experimenting with various ad-hoc combinations of layers we identify a custom CNN that for the specific problem we are considering and at least for the preliminary tests that we have run, outperforms the standard ones. Such a CNN, that we name AccCNN is shown in Figure 2. As we can see, the structure is not particularly complex. AccCNN first takes as input the image corresponding to the spectrogram of the accelerometer measurements, represented by a matrix $224 \times 224 \times 3$, and resizes it to a $32 \times 32 \times 3$ matrix (applying a *bilinear interpolation*). Then, through a sequence of three pairs of layers *Conv2D/MaxPooling2D* (with *relu* activation), one *dropout* of 0.2 produces a vector of 1024 elements, which is given as input to a *Flatten* layer. Such a layer is then fully connected to two *Dense* layers (size 128 and 64), in turn connected to an *Output layer* of size 51 (the number of speech units considered in our study).

3.2. Data collection

To collect the accelerometer signals measurements used to train the deep learning models, we have used a Samsung S8 smartphone. We wrote the SpyApp that records the accelerometer measurements. We have used the “table setting”, that is, during the experiment the smartphone is placed on a table. In this settings the accelerometer is solicited by the audio signal played

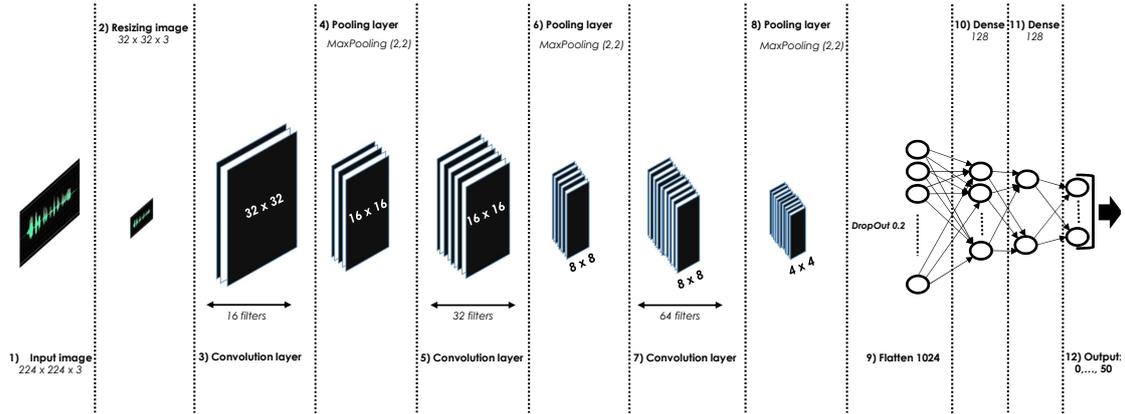


Figure 2: The AccCNN used for the speech units recognition.

through the loudspeaker of the device. As observed also in [9], acceleration signals collected from this setting show strong audio response along all axes. The speech units that we considered are the 10 digits plus the 21 letters of the Italian alphabet (Table 1).

| Speech units | Count |
|----------------------------|-------|
| Digits | 10 |
| Letters (Italian alphabet) | 21 |

Table 1
The 31 speech units

For each speech unit we have collected *samples*, that is accelerometer measurements during the reproduction of the speech unit. In total, we have collected 1200 samples for each speech unit. Since the number of speech units that we considered is 31, the total number of samples collected is $1200 \times 31 = 37200$.

3.3. Pre-processing

The goal of this phase is to find a representation for the accelerometer measurements that can be effectively learned by deep learning models. Given the accelerometer signals measurement in a time interval, the accelerometer measurements are transformed into a *spectrogram*. The spectrogram representation reflects the multi-scale information of a signal in the frequency domain, and it has been proved to be useful for widely adopted models in computer vision. The following describes the details of the transformation of the raw acceleration measurements into spectrograms.

1. *Interpolation*: in order to generate acceleration signals with a fixed sampling rate of 1000 Hz, (i) first, we used linear interpolation to deal with unstable intervals of accelerometer measurements, (ii) then, we upsampled the accelerometer measurements to 1000 Hz, and (iii) finally, we used timestamps to locate all time points that have no accelerometer measurement and used linear interpolation to fill in the missing data.

2. *High-pass filtering*: A high-pass filter has been used to eliminate (possible) significant distortions in the signals, and to obtain filtered signals mainly consisting of the target speech information and the self-noise of the accelerometer; specifically, we first convert the acceleration signal along each axis to the frequency domain using the Short-Time Fourier Transform (STFT), which divides the long signal into equal-length segments and calculates the Fourier transform on each segment separately; we then set the coefficients of all frequency components below the cut-off frequency (set to 80Hz to cover the adult males and females frequencies and to minimize the impact of noise components) to zero and convert the signal back to the time domain using inverse STFT.
3. *Signal-to-spectrogram*: since we have acceleration signals along three axes, three spectrograms can be obtained for each speech unit signal; to this, we first divide the signal into multiple short segments with a fixed overlap (as proposed in [9], we used 128 and 120 as signal and overlap lengths respectively); we then window each segment with a Hamming window and calculate its spectrum through STFT; the signal along each axis is now converted into a STFT matrix that records the magnitude and phase for each time and frequency. Finally, the 2D spectrogram can be calculated as $spect(s) = |STFT(s)|^2$, where s and $|STFT(s)|^2$ respectively represents a single-axis acceleration signal and the magnitude of its corresponding STFT matrix.
4. *Spectrogram-to-image*: to feed the spectrograms into the deep learning models chosen for our experiments, we convert the three 2-D spectrograms of a signal into one RGB image in PNG format; to this, (i) we fit the three $m \times n$ spectrograms into one $m \times n \times 3$ tensor, (ii) we take the square root of all the elements in the tensor and map the obtained values to integers between 0 and 255 (to obtain considerable information loss), (iii) we export the $m \times n \times 3$ tensor as an image in PNG format, (iv) the spectrogram-images are cropped to the frequency range from 80 Hz to 300 Hz in order to reduce the impact of selfnoise; (v) finally, to feed those images into standardized computer vision models, it is better to resize them into $n \times n \times 3$ images (to preserve sufficient information, usually $n = 224$).

3.4. Training, validation, and testing

Each of the model we have considered has been trained, validated and test as follows. As a first step, using the stratification on the set of labels, we partitioned the dataset of 37200 images into two subsets:

1. a *training set* with 80% of the images
2. a *testing set*, with 20% of the images.

Tables 2 and 3 show the results of the validation and testing experiments. The ad-hoc network that we propose, *AccCNN*, outperforms the other models with respect to all metrics, behaving slightly better than the DenseNet used in [9] and better than the other standard models. Specifically, in the validation phase the values achieved for *accuracy*, *precision*, *recall* and *f-score* are, respectively, 0.94, 0.91, 0.91 and 0.91, very close and slightly better than those of the model based on a DenseNet, while in the testing phase the values are 0.89, 0.88, 0.86, 0.86 again with a slight improvement over the Dense Net. The other models show worse performances, as can be seen from the tables.

It is worth to note that the improvement of the performance of AccCNN becomes more evident in the testing phase. Recalling that AccCNN has a much simpler structure (only 12 layers) this leads to the following observation: (at least) for the specific problem that we are considering and for the training set used in the training phase, a model with a simpler structure, such as AccCNN, can achieve a higher generalization capacity on the testing set. The reason why AccCNN works better than the others lies in the *over-parametrization* issues affecting DNNs. Often, the choice of very complex models does not necessarily ensure better performance. Models with many parameters, such as pre-trained CNNs, have a high capability to fit the noise at the expense of a lower generalization capacity. This is especially evident when the representations used for the samples are not sophisticated enough. In general, more complex models would probably have needed more data or somehow more sophisticated representations.

| Model | Accuracy | Precision | Recall | F-score |
|------------|----------|-----------|--------|---------|
| VGG | 0.87 | 0.83 | 0.83 | 0.85 |
| ResNet | 0.87 | 0.85 | 0.84 | 0.84 |
| WideResNet | 0.87 | 0.84 | 0.84 | 0.86 |
| DenseNet | 0.92 | 0.91 | 0.89 | 0.91 |
| AccCNN | 0.94 | 0.91 | 0.91 | 0.93 |

Table 2

Performance in the *validation* phase: letter + digits.

| Model | Accuracy | Precision | Recall | F-score |
|------------|----------|-----------|--------|---------|
| VGG | 0.75 | 0.73 | 0.73 | 0.70 |
| ResNet | 0.75 | 0.70 | 0.70 | 0.71 |
| WideResNet | 0.79 | 0.77 | 0.76 | 0.78 |
| DenseNet | 0.86 | 0.83 | 0.86 | 0.86 |
| AccCNN | 0.89 | 0.88 | 0.86 | 0.86 |

Table 3

Performance in the *testing* phase: letter + digits.

4. Generalization

The speech units recognizer that we have described in the previous section has the specific goal of identifying single digits or single letters. Clearly it is desirable to generalize the recognizing capabilities of the system to words or even sentences. As much clearly is the fact that the task is not easy. In [3] beside the single digit and the single letters a small set of “hot” keywords has also been considered, namely “password”, “username”, “social”, “security”, “number”, “email”, “credit”

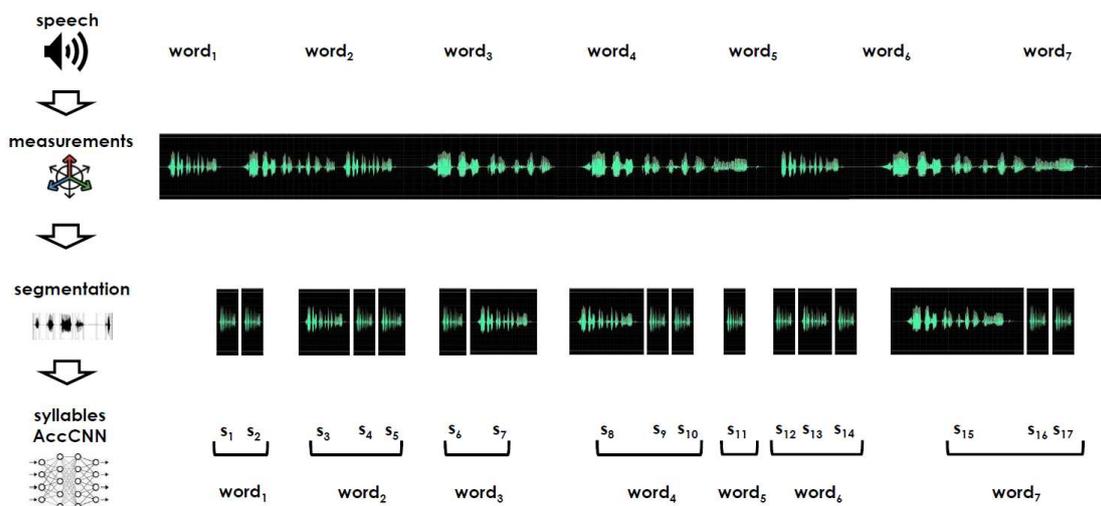


Figure 3: The steps of the proposed approach for recognizing entire words.

and “card”. Adding new words requires re-building the model and reaching a rich enough set can be quite difficult. Instead of targeting a specific set of words we propose an alternative approach: build a speech units recognizer for the syllables and then use design segmentation techniques to identify sequences of syllables corresponding to actual words. As we can see in Figure 3, given the accelerometer measurements corresponding to a sequence of words pronounced during a conversation, the proposed strategy consist of the following steps: (i) a segmentation technique is applied to extract the measurements corresponding to the syllables composing the words, (ii) each extracted measurement is given as input to AccCNN, (iii) the recognized syllables are assembled to reconstruct the original words.

As a first step we have considered a set of “dummy” syllables: all the ones that can be obtained appending a vowel to the consonants b, d, r and s , namely: $\{ba, be, bi, bo, bu, da, de, di, do, d, ra, re, ri, ro, ru, sa, se, si, so, su\}$. Of course this means that also the “words” that we will consider are “dummy” words (although a few meaningful words can be constructed with the syllables that we are considering). We have this simplifying assumption in order to have a small set of similar syllables to understand whether the approach can actually work. It goes without saying that the approach needs to be expanded to consider the set of all possible (real) syllables.

Having established set of syllables the next step is that of taking an entire conversation and identify the syllables use in order to check for specific sequence of consecutive syllables that make up words. In order to do so we need to face the problem of “segmenting” the entire conversation into pieces that correspond to the syllables. We explore a simple approach: dividing the entire conversation into small pieces each one corresponding to a syllable. Syllables have different length so it is not clear how the conversation should be split into pieces. We tried with a very simple approach: use segments of the same length and as the length, using 5 different lengths (0.50, 0.55, 0.60, 0.65, 0.70).

In order to consider the syllables we have to train the network on the syllables. So we repeated the training, validation and testing phases described in Section 3 considering the chosen 20

syllables instead of the digits and the letters of Table 1.

Tables 4 and 5 show the results of the validation and testing experiments. The results are similar to those obtained for the digits and letters (Tables 2 and 3). It is possible to notice that in this case the performance of AccCNN and DenseNet are almost the same.

| Model | Accuracy | Precision | Recall | F-score |
|-------------------|-----------------|------------------|---------------|----------------|
| <i>VGG</i> | 0.88 | 0.84 | 0.85 | 0.85 |
| <i>ResNet</i> | 0.88 | 0.84 | 0.86 | 0.86 |
| <i>WideResNet</i> | 0.88 | 0.87 | 0.86 | 0.87 |
| <i>DenseNet</i> | 0.93 | 0.91 | 0.91 | 0.90 |
| <i>AccCNN</i> | 0.95 | 0.91 | 0.91 | 0.92 |

Table 4

Performance in the *validation* phase: syllables.

| Model | Accuracy | Precision | Recall | F-score |
|-------------------|-----------------|------------------|---------------|----------------|
| <i>VGG</i> | 0.83 | 0.73 | 0.76 | 0.76 |
| <i>ResNet</i> | 0.85 | 0.77 | 0.75 | 0.76 |
| <i>WideResNet</i> | 0.86 | 0.80 | 0.81 | 0.82 |
| <i>DenseNet</i> | 0.90 | 0.87 | 0.87 | 0.88 |
| <i>AccCNN</i> | 0.90 | 0.88 | 0.87 | 0.88 |

Table 5

Performance in the *testing* phase: syllables.

To test the recognizer we have used a set of 100 “sentences” of varying length, from 5 to 60 seconds (roughly 8 for each length). Each sentence is simply a sequence of (dummy) words built with the dummy syllables, using 2, 3 or 4 syllables per word. An example of a 5-second “sentence” is *dodababe dore babesa* and an example of a 25-second sentence is *babada direro doredo disa sasasesa bubiduda da da sese babababi bibi suso sasasasa siredomi dada*

Since to use the recognizer we need to segment the sentences, we have to decide the length of the segments. To do so, we analyzed the lengths of the 24000 samples of the speech units (1200 per each of the 20 speech units): they range (roughly) from 0.5 to 0.7 seconds. Thus we tried to segment the sentences with the following values: 0.50, 0.55, 0.60, 0.65 and 0.70 seconds.

Figure 4 shows the results in terms of percentage of recognized words as a function of the length of the sentence and of the length of the segments. The recognizer seem to perform badly with very short sentences (of 5 and 10 seconds) and better with longer ones (15 to 45 seconds); the performance tends to degrade for very long sentences (50 to 60 seconds). Moreover the segmentation with segments of 0.55 seconds seems to be the one that works better. Overall the percentage of words recognized is low, always less than 35%. This is probably due to the

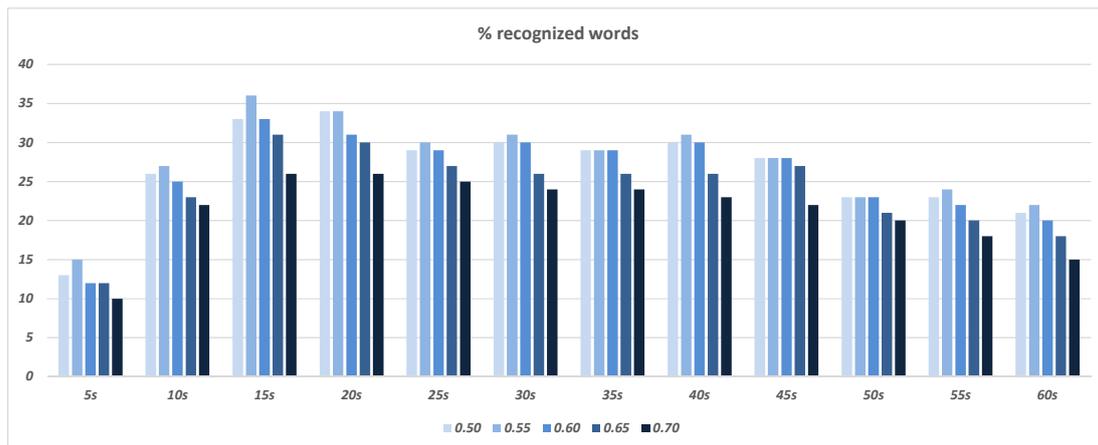


Figure 4: Percentage of recognized words as a function of the length of the sentences (from 5s to 60s) and for each length as a function of the length of the segments (from 0.50s to 0.70s)

fixed segmentation which does not capture correctly the single syllables. A clever and more sophisticated approach needs to be used.

5. Better segmentation strategies

The segmentation approach presented in the previous section is quite naive; it would be interesting to design a more “intelligent” segmentation technique. A first step in this direction could be that of (i) defining a *metric* for measuring how good a segmentation is, i.e., how much it consists of segments that can be “effectively” classified by AccNN, and then (ii) to show that this metric is correlated to the number of words correctly recognized in the conversation.

Assessing the goodness of the segmentation In order to define such a metric one could exploit the fact that an AccCNN computes the probability that an image in input “belongs” to one of the 51 classes (speech units). As a step in this direction we performed a preliminary analysis, in which we have analyzed these probabilities on the images in both training and testing set defined (Section 3.4), and on images extracted at random from recorded conversations. As a result, we observed that images corresponding exactly to speech units are always characterized by only one probability close to 1 (corresponding to the speech unit in which the image will be classified by AccCNN) while the others close to 0. Conversely, in the case of images that do not correspond precisely to speech units, such a distinction is not so evident. This observation can help in defining a measure for the goodness of a segmentation. Given a segmentation (list of segments): (i) for each segment one can consider the list of probabilities of belonging to each class (51 values) and can compute the “value” of such a segment, i.e., the difference between the highest value and the average sum of the remaining values, then (ii) then it is possible to compute the “value” of the entire segmentation as the average sum of the values of its segments.

Statistical relation with the capability of recognizing words. The successive step that we plan to take is that of studying the correlation between the segmentation quality metric and the number of words correctly recognized in speech conversations. In order to do so we plan to do the following: (i) use the table setting described in Section 3.2 to collect the acceleration signals measurements corresponding to a number of conversations (as combination of speech units) of variable length; (ii) for each conversation generate random segmentations; (iii) give each segmentation obtained as input to AccCNN (one segment at a time) and then count the number of words correctly recognized; (iv) for each segmentation compute the segmentation quality and the number of words correctly recognized, (v) study the correlation between such two distributions of data by using the Shapiro-Wilk goodness-of-fit test ([17]) to assess the normality of the data (the non-normality of distributions led us to apply the well-known non-parametric Spearman's rho test).

Strategies to find good segmentation. Once we have defined the above cited metric and have proved that it is actually a good metric for identifying the segmentations that allows us to obtain the best recognition of the words we plan to exploit it to define algorithms that allow to identify such segmentations. We believe that genetic algorithms could be effective.

6. Conclusion

In this paper we have tackled the problem of inferring private information exploiting the accelerometer of a smartphone by measuring the vibrations caused by speeches reproduced on the device itself. We have designed an approach based on deep-learning methods and assessed its behavior through experimental data. The system is designed for recognizing single letters and single digits. We have also designed a generalization to words based on the recognition of the syllables providing preliminary results. Although the applicability of the proposed system seems restricted, and the generalization technique is still embryonic, the results obtained are interesting and suggest future directions to follow in order to improve the effectiveness of the attack. As future work we plan to study more in details the proposed generalization to words, as explained in Section 5. The study presented in this paper uses only a small set of dummy syllables. It would be interesting to expand this set to include all the syllables in a given language and then try to recognize words in real conversations. The current proposed approach uses quite a straightforward (and not very clever) method to segment a long conversation into units that correspond to syllables. It would be interesting to study better ways to performing the segmentation, for example using dynamic approaches that could adapt to the syllables that have been recognized.

References

- [1] Y. Chen, C. Shen, Performance analysis of smartphone-sensor behavior for human activity recognition, *IEEE Access* 5 (2017) 3095–3110.
- [2] C. Shen, Y. Chen, G. Yang, On motion-sensor behavior analysis for human-activity

- recognition via smartphones, in: 2016 Ieee International Conference on Identity, Security and Behavior Analysis (Isba), IEEE, 2016, pp. 1–6.
- [3] E. Davarci, B. Soysal, I. Erguler, S. O. Aydin, O. Dincer, E. Anarim, Age group detection using smartphone motion sensors, in: 2017 25th European Signal Processing Conference (EUSIPCO), IEEE, 2017, pp. 2201–2205.
- [4] S. Majumder, M. J. Deen, Smartphone sensors for health monitoring and diagnosis, *Sensors* 19 (2019). URL: <https://www.mdpi.com/1424-8220/19/9/2164>. doi:10.3390/s19092164.
- [5] A. Sharshar, A. Fayez, Y. Ashraf, W. Gomaa, Activity with gender recognition using accelerometer and gyroscope, in: 2021 15th International Conference on Ubiquitous Information Management and Communication (IMCOM), IEEE, 2021, pp. 1–7.
- [6] L. Hernández-Álvarez, J. María de Fuentes, L. González-Manzano, L. H. Encinas, Smart-campp - smartphone-based continuous authentication leveraging motion sensors with privacy preservation, *Pattern Recognition Letters* (2021).
- [7] A. Al-Haiqi, M. Ismail, R. Nordin, On the best sensor for keystrokes inference attack on android, *Procedia Technology* 11 (2013) 989–995. doi:<https://doi.org/10.1016/j.protcy.2013.12.285>, 4th International Conference on Electrical Engineering and Informatics, ICEEI 2013.
- [8] S. A. Anand, N. Saxena, Speechless: Analyzing the threat to speech privacy from smartphone motion sensors, in: 2018 IEEE Symposium on Security and Privacy (SP), IEEE, 2018, pp. 1000–1017.
- [9] Z. Ba, T. Zheng, X. Zhang, Z. Qin, B. Li, X. Liu, K. Ren, Learning-based practical smartphone eavesdropping with built-in accelerometer, in: *Proceedings of the Network and Distributed Systems Security (NDSS) Symposium*, 2020, pp. 23–26.
- [10] P. Marquardt, A. Verma, H. Carter, P. Traynor, (sp) iphone: Decoding vibrations from nearby keyboards using mobile phone accelerometers, in: *Proceedings of the 18th ACM conference on Computer and communications security*, 2011, pp. 551–562.
- [11] Y. Michalevsky, D. Boneh, G. Nakibly, Gyrophone: Recognizing speech from gyroscope signals, *SEC'14*, USENIX Association, USA, 2014, p. 1053–1067.
- [12] L. Zhang, P. H. Pathak, M. Wu, Y. Zhao, P. Mohapatra, Accelword: Energy efficient hotword detection through accelerometer, in: *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services*, 2015, pp. 301–315.
- [13] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, in: Y. Bengio, Y. LeCun (Eds.), *3rd International Conference on Learning Representations, ICLR 2015*, San Diego, CA, USA, May 7-9, 2015, 2015.
- [14] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.
- [15] S. Zagoruyko, N. Komodakis, Wide residual networks, in: E. R. H. Richard C. Wilson, W. A. P. Smith (Eds.), *Proceedings of the British Machine Vision Conference (BMVC)*, BMVA Press, 2016, pp. 87.1–87.12.
- [16] G. Huang, Z. Liu, L. Van Der Maaten, K. Q. Weinberger, Densely connected convolutional networks, in: *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.
- [17] S. S. Shapiro, M. B. Wilk, An analysis of variance test for normality (complete samples), *Biometrika* 52 (1965) 591–611.