Automatic driver phone hand-usage detection: a cepstrum-based approach

Simone Gelmini, Simone Formentin, Silvia Strada, Mara Tanelli*, and Sergio Savaresi

Abstract— **In the last few years, smartphone usage has been unanimously considered to be one of the most dangerous driving habit, leading to numerous road accidents. In the context of insurance telematics, the online estimation of the driving style is of utmost importance to offer personalized policies and to evaluate riskiness levels. Smartphones can be used to gather dynamics measures needed to build riskiness indicators to evaluate drivers' behaviour. Being able to recognize phone usage is of paramount importance. This work deals with the problem of detecting the use of the phone during a trip. To do this in a fully automatic fashion, we propose a new cepstrum-based classification method based on time series analysis. The resulting performance is tested on experimental data and compared with those obtained with another method, based on hand-crafted features, showing that the new approach yields fully comparable performance while significantly increasing the automation level of the classification process.**

I. Introduction

It is nowadays widely acknowledged that the use of mobile devices while driving contributes to a significant amount of fatalities [1]. Because smartphone usage is strongly correlated with the occurrence of accidents, clearly insurance companies are very much interested in being able to estimate this condition, in order to shape their policy offers based on this trait of their clients, [2], [3]. In turn, being offered discounts from the insurance companies in exchange for phone-free driving could be a really persuasive means to help spreading this desirable habit. To do this, detecting whether and how often a phone is used when actively driving is central.

It is worth noting that, differently from what happens for traditional driving style profiling for safety-oriented purposes, where vehicle's dynamic variables (*e.g.*, speed and acceleration) are monitored for the final assessment, [4], [5], in this case one must gather information on the smartphone motion itself. In the existing literature, little or no effort has been devoted up to now to detect the use of the phone while driving, which can be due to texting, making a phone call or simply accessing the Internet.

To the best of our knowledge, such problem has been considered only in few contributions. In [3], phone usage recognition is accomplished in a quite naïve fashion, *i.e.*, by simply looking if significant acceleration or angular velocity changes are sensed by the phone sensors within a short time-window. This enables checking if a pick-up or drop-off of the phone occurred, but not if the device has been really used in between. In [6], integrated inertial sensors are employed, detecting the simultaneous occurrence of driving and texting in real-time, by mining a well defined pattern. In [7], the phone usage detection system leverages the existing car stereo infrastructure, in particular the speakers and the bluetooth network. The same problem has been analyzed also by [8], [9], where the detection of the phone is obtained by the fusion of the smartphone built-in sensors and external devices.

We have started considering this problem in [10], where we approached it as a binary classification task on the time series measured by the smartphone's sensors, solved using a Support Vector Machine (SVM) method. A similar attempt to classify instances of distracted driving due to smartphone usage is presented in [11]. In this paper, significant features are extracted from smartphone sensors and classified by means of a random-forest-based algorithm. The limitation of this study, which ours overcomes, is that all the experiments were conducted in a closed simulation environment, where, for example, real roads effects and other practical issues are not considered.

Even though the method we presented in [10] was indeed based on experimental data and proved feasible with good performance, it heavily relied on a strong knowledge of the application domain (as all the other mentioned approaches do), and it entailed a pre-processing phase and a feature selection method resting on subjective reasoning, which is terribly time-consuming. Therefore, to offer a full automation of the classification task, in this work we propose a novel approach for the smartphone usage detection, based on the computation of cepstrum coefficients. Although the cepstrum was first defined in 1963 [12], its use for time series clustering and modeling in dynamical systems has been only recently prompted by De Moor and coauthors, see, *e.g.*, the recent [13].

Specifically, the classification is solved using the time series measured by the smartphone IMU and the results compared to those obtained with the previously mentioned SVM-based classifier presented in [10]. To our best knowledge, this is the first work where cepstrum-based signal processing is used for activity detection *via* classification. The paper shows that the cepstrum-based approach yields results which are definitely comparable to the best SVM classifier, with the significant advantage of avoiding the subjective and time-consuming classifier design phase (Fig. 1).

The paper is organized as follows: Section II defines the

The authors are with the Dipartimento di Elettronica Informazione e Bioingegneria, Politecnico di Milano, via G. Ponzio 34/5, 20133, Milan, Italy. Email: {simone.gelmini | simone.formentin | silvia.strada | mara.tanelli | sergio.savaresi}@polimi.it

* Corresponding author

problem and the experimental setup, while III presents the cepstrum-based classification approach. Section IV illustrates the results obtained on the considered application, showing the effectiveness of the proposed solution.

II. Problem statement and experimental setup

In this paper, a cepstrum-based smartphone use mode detection algorithm is proposed. The purpose of the algorithm is to detect whether a driver is using the phone while driving, based only on the phone's sensors measures. The algorithm is automatically activated and deactivated when the driver is inside the vehicle, thanks to a Bluetooth beacon placed under the steering wheel. In presence of multiple smartphones, the beacon activates only the closest device. A fine detection whether the activated phone belongs to the driver or to a passenger is out of the scope of this work.

The smartphone is considered *in-use* when the driver performs one of the most common activities (e.g., handling, texting, scrolling, browsing, calling etc.) and *not in-use* when the phone is on the phone holder or on the passenger seat.

The experimental setup is composed of a smartphone – equipped with a tri-axial gyroscope unit – and the activating beacon. Data are recorded at the maximum sampling frequency of 120 Hz. The designed classifier performs a high rate classification. However, as the phone detection is used for profiling the tendency to use the phone while driving, the final output will be averaged, reducing the classification to 1 Hz.

III. Cepstrum based classification

In the proposed contribution, a cepstrum-based classifier for detecting the use of the smartphone is presented. The methodology and the theoretical background are introduced and briefly discussed in this section.

A. What is the cepstrum?

According to [12], the fathers of cepstrum defined it as "the power spectrum of the logarithm of the power spectrum". In fact, given a signal $s(t)$, its cepstrum coefficients c_s are defined as the inverse Fourier transform of the logarithm of the spectrum Φ_s [15], as

$$\begin{aligned} c_s(k) &= \mathcal{F}^{-1}\left[\log\left(\Phi_s\right)\right] \\ &= \frac{1}{2\pi}\int_0^{2\pi} \log\left(\Phi_s\left(e^{i\theta}\right)\right) e^{ik\theta} d\theta. \end{aligned} \tag{1}$$

The denomination *cepstrum* was introduced by the authors of [12], paraphrasing terms like spectrum (cepstrum), frequency (quefrency) - and many more - in order to avoid confusion with the meaning of the original ones, as described in [16].

Cepstrum coefficients have been widely used for time series clustering. In time series clustering, it is not obvious how to find similarities in temporal data. With cepstrum, time series are grouped based on their similar dynamics, making the clustering process effective. Clusters are formed comparing the cepstrum coefficients by means of suitable metrics and grouping together instances that are neighbors.

Numerous cepstrum-based metrics have been proposed to solve this problem, [13]. However, the most used is the so-called Martin distance [17]: given two time series generated by two different single input-single output (SISO) linear time-invariant (LTI) autoregressive-moving average (ARMA) models $M_1,\ M_2$, the Martin distance is defined as

$$d(M_1, M_2) = \sqrt{\sum_{k=0}^{\infty} k|c_1(k) - c_2(k)|^2}, \tag{2}$$

where $c_1,\ c_2$ are the cepstrum coefficients associated to output of the models M_1 and M_2, respectively. This metric has become widely popular because it is easy to calculate and significantly reduces the computational effort with respect to other ones [13].

B. Training and classification algorithms

In the proposed work, cepstrum coeffiecients are employed in order to solve a classification problem, thanks to their high capabilities to discriminate similar patterns based on the differences in the time series spectra, as shown in [18], [19]. In this subsection, the training (run once offline) and classification (which can be run both offline and in real-time) algorithms are described, separately.

1) Training: Training is performed on the set of time series recorded during the tests. More specifically, as illustrated in Section II, smartphone's sensors measure three angular rates. To generalize the patterns and being robust with respect to the orientation of the smartphone, the Euclidean angular rates norm is derived

$$\|\omega\| = \sqrt{\omega_x^2 + \omega_y^2 + \omega_z^2}, \tag{3}$$

so that a single inclusive signal represents the intensity of all the angular rates together. The choice of using (3) is not random, but it proved to be the most informative signal for this application in a previous work of the authors [10].

Each time series contains a different pattern to be mined. Eventually, multiple patterns may belong to the same class (*e.g.*, in this application, *on passenger seat* and *on phone holder* belong to the same *not in use* class).

Goal of the training phase is to create a database of known dynamics that are used for comparison during classification. This means that, for each time series, the cepstrum coefficients are computed. To this end, given y_{cl_p}, a time series p belonging to class cl, its spectrum Φ_{cl_p} is computed on a sliding window of dimension $N = \frac{\alpha}{T_s}$ - where α is the dimension of the window expressed in seconds and T_s is the sampling time - by means of the Fast Fourier Transform (FFT). The sliding window is needed to regularize the spectrum and to compute the cepstrum coefficients with the same framework used in the classification phase, which has to be run real-time with an incoming stream of data in a limited size buffer. Mathematically, this corresponds to:

$$\bar{\Phi}_{cl_p} = \frac{1}{T}\sum_{t=0}^{T} \Phi_{cl_p}(t), \tag{4}$$

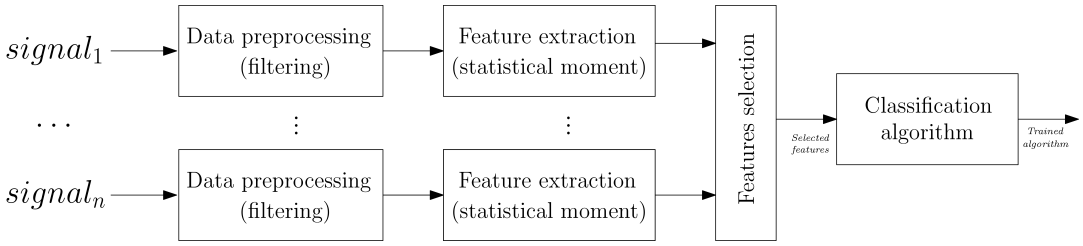


Fig. 1: A schematic of the learning process of many applications: measured signals measured are initially preprocessed; then, features are extracted and combined maximizing the classification performance. These steps may require deep knowledge of the problem and, eventually, time-consuming calibration. *Can this process be automated instead?*

Fig. 2: The experimental setup: the smartphone (a OnePlus One [14]) and the Bluetooth beacon, pleaced under the steering wheel.

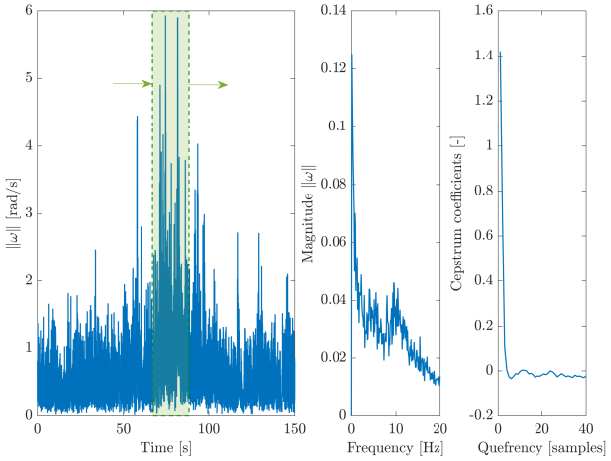


Fig. 3: Given a moving window, the spectrum of the signal is computed on the buffered data. Then, cepstrum coefficients are evaluated from the computed spectrum.

in which T is the length of the time series vector. Cepstrum coefficients are then evaluated on the logarithm of the regularized spectrum $\bar{\Phi}_{cl_p}$ through the Inverse Fast Fourier Transform (IFFT).

This set of operations is repeated for all the time series available for learning. The obtained coefficients are used to create the set of learned patterns.

2) Classification: Classification is performed on data buffered in a sliding window, with the same length of the training one. At each time step, the spectrum and then, consequently, the cepstrum coefficients are computed, as shown in Fig. 3.

The computed coefficients are used to compare the dynamics of the current data with respect to the learned baseline by means of the Martin distance. However, the coefficients vector is not infinitively long and, more importantly, decays to zero rapidly (Fig. 3), as proved also by [20]. Furthermore, the estimation of high-order cepstrum coefficients may be difficult and easily corrupted by noise, as it is widely known for the autocovariance case [15]. For this reason, computing the Martin distance on infinite terms (or even with a great number of them) is impractical or results may be unacceptable. In this paper, we propose to evaluate the metric only on the first N_{cep} coefficients, modifying (2) as follows

$$d(c_{online}, c_{cl,p}) = \sqrt{\sum_{k=0}^{N_{cep}} k|c_{online}(k) - c_{cl_p}(k)|^2}, \quad (5)$$

in which N_{cep} can be chosen based on prior knowledge on the dynamic behavior of the system or, like in this application, as a tuning parameter; instead, c_{online} and $c_{cl,p}$ are, respectively, the cepstrum vectors of the online data and of the training data of a specific time series p belonging to class cl.

The predicted class $\widehat{Y}$ is the one minimizing the distance in (5):

$$\widehat{Y} = arg \min_{cl \in \mathcal{CL}} \min_{p \in \mathcal{P}} d(c_{online}, c_{cl_p}), \quad (6)$$

with $\mathcal{CL}$ the set of classes and $\mathcal{P}$ the set of patterns for each class in $\mathcal{CL}$. An example of one step of classification is shown in Fig. 4, in which the algorithm performs the classification on a previously recorded stream of data.

IV. Application and Results

In this context, the classifier is binary (classes *in-use* and *not in-use*), meaning $|\mathcal{CL}| = 2$. On the contrary, $|\mathcal{P}| = 1$ for class *in-use* and $|\mathcal{P}| = 2$ for class *not in-use* (scenarios *on phone holder* and *on passenger seat*).

In Fig. 5, the spectra of $\|\omega\|$ in three different scenarios is analyzed, both when the vehicle is in motion and standing still. The spectrum of *using* is quite different from *on passenger seat* and *on phone holder* over the all frequency range. This difference is even more accentuated when the vehicle is standing still because measurements are no longer influenced by the vehicle dynamics, as when the vehicle is

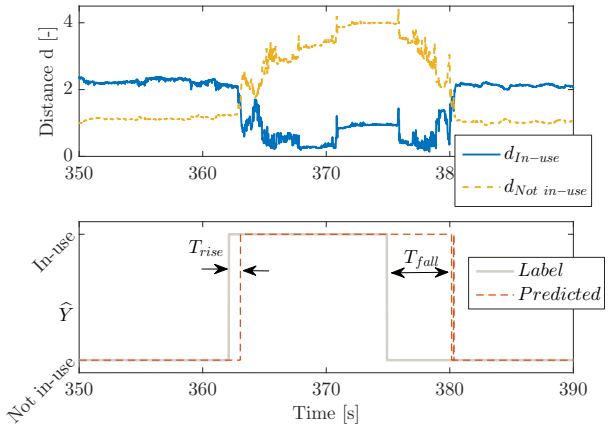


Fig. 4: An example of the classification process. Top: the Martin distance computed for all the classes involved. Bottom: the predicted output based on the computed distance. The classification transients are noted with T_{fall} and T_{rise}.

in motion, with peaks in the magnitude up to 3 Hz. These differences in the spectra can be used as a signature to discriminate when the driver is using the phone, which can be easily captured computing the cepstrum coefficients of the time series associated to these events.

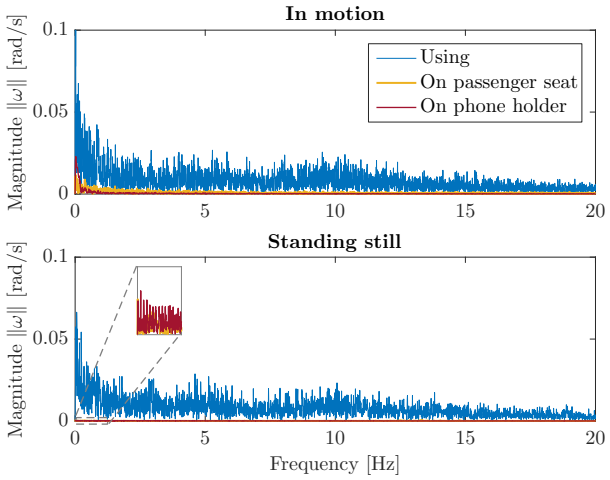


Fig. 5: Spectra of the analyzed scenarios when the vehicle is in motion (on the top) and standing still (bottom plot). In *using*, the magnitude of the spectrum is greater than otherwise, in both the driving conditions.

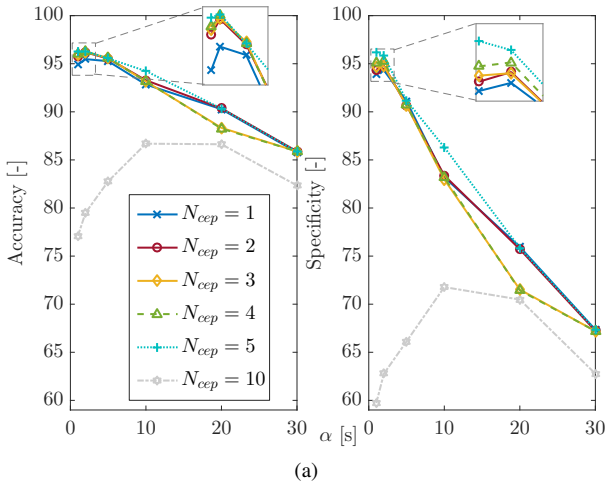
The classification algorithm presented in the previous section is tested against data collected during an experimental campaign conducted in mixed driving conditions (*i.e.*, mixed urban and highway) in which the phone has been used according to the real behavior of the user (*i.e.*, mixed in-use and not-in-use scenarios of different temporal length).

Since the proposed classification algorithm has two tuning parameters (N_{cep}, the maximum cepstrum order, and α, the window length), a sensitivity analysis on the validation dataset is performed, in terms of accuracy, specificity and sensitivity (Fig. 6) indexes computed as in [21]. The most sensitive parameter is α. In fact, after an initial increase of accuracy for small windows ($\alpha \leq 5$ s), the performance index shows a significant drop for larger ones. *Why?* The value of α affects how long the transient[1] is: with a greater window, more time is required to detect a change of class in the input. According to the specificity and sensitivity trends, the effect of the transients is more evident in detecting when the driver stops using the phone than the dual condition. This is rather intuitive: When the user stops using the phone, the signal stored in the buffer is still influenced by the previous instants in which the signal contains more high frequency components, as shown in Fig. 5; the buffer needs to empty those harmonics to be able to detect the right pattern. In the dual condition, the classification algorithm is prompt at detecting those as soon as the buffer gets filled of enough discriminating components. In fact, with great values of α, the FFT becomes finer, leading to an increment of sensitivity, which allows to obtain a more accurate classification than when the driver starts using the phone. Furthermore, although less relevant, another positive effect of a wider window consists in a finer classification in terms of spikes. Spikes can be defined as very short-time outliers in the classification, too short to be realistic according to the involved dynamics (*e.g.*, in this application, spikes are false positives and false negatives shorter than 0.2 s, usually not longer than few samples – it is very unlikely to be able to robustly detect such a short use of the phone). As shown in Fig. 6, the number of spikes is decreasing for longer windows. Instead, the order of the cepstrum coefficients (N_{cep}) proves to be a less critic parameter for small values, but performance drops significantly for $N_{cep} > 5$, according to all the performance indexes.

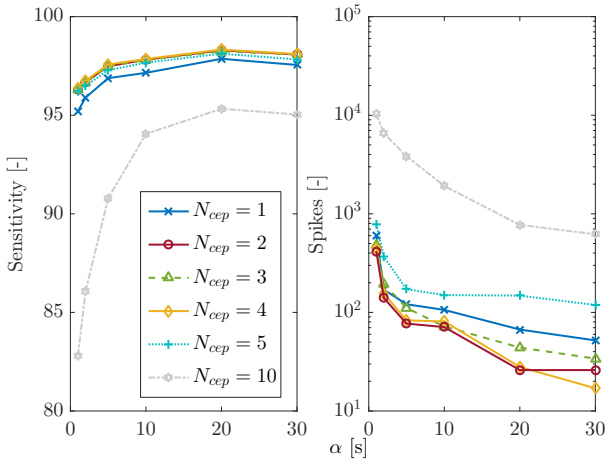
In our case, a good trade-off in the classification performance is obtained with $\alpha = 5\ s$ and $N_{cep} = 3$. With these parameters, the classifier achieves an accuracy of 95.6%, sensitivity reaches 97.68%, specificity is 90.68%, and 83 spikes are counted. However, for a fair performance evaluation, the tuned classifier is tested against a different dataset. The test dataset is chosen to be different from the validation one, with a shorter, but more frequent, use of the phone. In this case, classification performance slightly drops in terms of accuracy (which is now 91.96%) and specificity (which reaches 74.52%); a limited variation is registered for sensitivity (97.15%) and in the count of the spikes (122).

To understand this performance drop, the effect of transients on the classification is analyzed. *Steady* denotes the mean time between the variation of the label and when the output reaches a steady classification, an index of the averaged duration of transients. As shown in Fig. 7, the classifier is generally faster at detecting when the user begins to use the phone (denoted as rise time) than the dual condition (termed fall time), no matter the cepstrum order used. This is consistent with what we already discovered

[1]A transient is the classification inertia in detecting a change of class. In case of static data, since they do not have transients, the classification do not show any inertia. On the contrary, when dynamic, the effect of time on data is not negligible and influences the classification and its performance.



(a)



(b)

Fig. 6: Sensitivity analysis of the performance during validation based on the two tuning parameters: α and N_{cep}. Accuracy and specificity drop for windows longer than 5 s due to the long transients (top), while the number of spikes decreases (bottom). Performance drops both with $N_{cep} = 1$ and with $N_{cep} \geq 5$.

analyzing the sensitivity and specificity trends. Moreover, this proves that the effect of the transients is only due to the parameter α, which must be necessarily tuned according to the involved dynamics.

By removing the averaged transients from data and recomputing the performance indexes, the classifier is analyzed as it were constantly at steady state, when the buffer does not store data of two patterns and, then, the values of cepstrum are more consistent. Not surprisingly, the classifier outperforms the previous case (Fig. 8). The classifier found during the sensitivity analysis (with $\alpha = 5$ and $N_{cep} = 3$) is still the best performing also at steady state. In fact, accuracy reaches 98.67%, sensitivity is 98.39% and specificity jumps to 99.43%. A significant increase of the performance is found also in testing: accuracy moves from 91.96% to 96.93%, though sensitivity is almost constant (reaches 97.52%, when it is 97.15% in the nominal condition). The great improvement in accuracy is due to the significantly better specificity, which is 94.46%, almost 20 points more from 74.52%. Moreover, the classifier improves its performance for greater values of α, proving once more that the drop of accuracy and specificity was only due to the presence of the transients, and larger windows provide a more accurate classification. Transients are indeed a negative side effect of the classification which cannot be compensated otherwise, no matter what classification algorithm is used, as proved in a previous work of the authors [10].

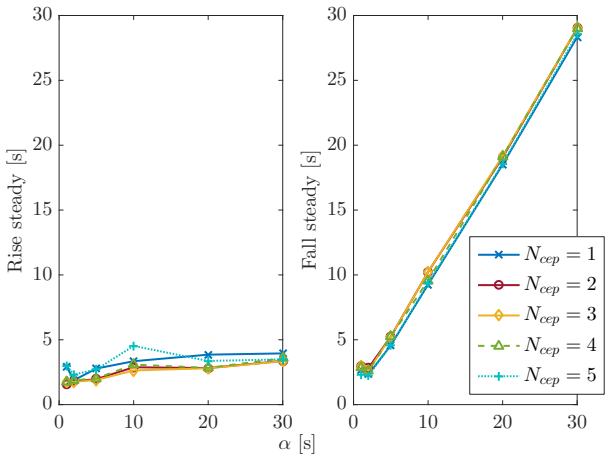


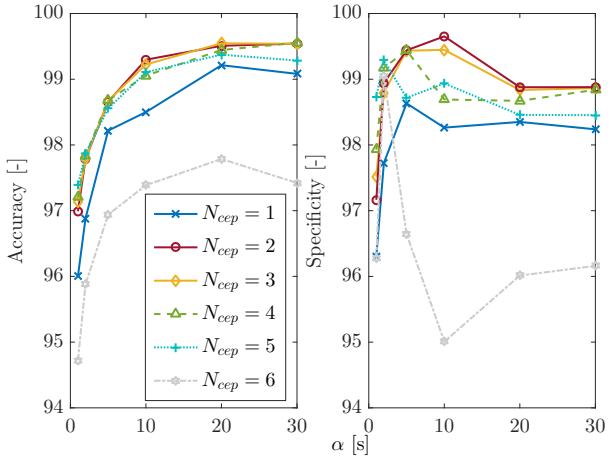
Fig. 7: The averages length of the transients is evaluated for different values of α and N_{cep}. Results prove that the classifier is not influenced by the tuning values at detecting when the driver begins to use the phone; instead, transients of the dual scenario are influenced by the window length.

In Table I, obtained results are compared with the different approach proposed in [10], in which an SVM algorithm was used. As shown, the cepstrum-based algorithm, compared to SVM, provides satisfying results, though less performing. However, the presented classification algorithm significantly reduces the data preprocessing and feature extraction phases, which can be time-consuming, as no general guidelines are available. Thus, this algorithm is recommended for all the applications in which the user does not have prior knowledge on the system or calibration effort is required to be minimal.

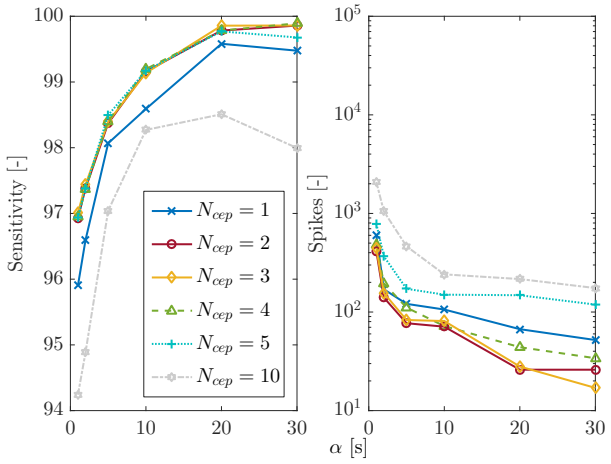
Furthermore, exactly as many other learning algorithms, also SVM has a parameter to be tuned. This is known as the capacity constant, which has to be properly tuned in order to avoid overfitting. In this application, for high values of the capacity constant, algorithm performance are settled and can be used as a benchmark to evaluate the proposed algorithm. However, when the parameter is chosen to be small enough, the training phase does not converge with that set of features, making the most performing classifier not usable and the comparison between SVM and the proposed algorithm unfeasible. Thus, the cepstrum-based classification allows to automate the part of the algorithm subject to manual design and tuning, with benefits also in the classifier tuning phase.

Dataset	Classifier	Features	Accuracy		Sensitivity		Specificity		Time rise steady [s]	Time fall steady [s]	Spikes
			Nominal	Steady	Nominal	Steady	Nominal	Steady			
Validation	SVM	$\|\omega\|_{bpf}$	96.38	99.74	99.19	99.95	89.44	99.17	1.27	6.36	1
	Cepstrum	$\|\omega\|$	95.60	98.67	97.57	98.39	90.68	99.43	1.89	5.23	83
Testing	SVM	$\|\omega\|_{bpf}$	93.05	98.42	98.11	98.47	76.43	98.21	0.25	4.87	1
	Cepstrum	$\|\omega\|$	91.96	96.93	97.15	97.52	74.52	94.46	0.26	4.39	122

TABLE I: Comparison of the performance of the better classifiers against validation and testing data.



(a)



(b)

Fig. 8: Sensitivity analysis of the performance during validation at steady state: α and N_{cep}. A finer and more robust classification is obtained with higher values of α, though performance settle for $\alpha \geq 5$, proving how detrimental the effect on transients is on the classification performance.

V. Concluding Remarks

This paper showed how a cepstrum-based approach can be effectively used to solve a classification problem using time-series data. This is done in a real application context, in which data coming from the inertial measurement unit on board of a smartphone are used as inputs. The obtained results favorably witness the promising performance of the approach, comparing it to a more classical SVM-based classification.

References

[1] S. Singh, "Distracted driving and driver, roadway, and environmental factor," *NCSA Report*, 2010.

[2] J. Wahlström, I. Skog, and P. Händel, "Driving behavior analysis for smartphone-based insurance telematics," 2015.

[3] F. Li, H. Zhang, H. Che, and X. Qiu, "Dangerous driving behavior detection using smartphone sensors," *Proceedings of the 19th International Conference on Intelligent Transportation Systems*, 2016.

[4] D. A. Johnson and M. M. Trivedi, "Driving style recognition using a smartphone as a sensor platform," *Proceedings of the 14th International Conference on Intelligent Transportation Systems (ITSC)*, 2011.

[5] H. Eren, S. Makinist, E. Akin, and A. Yilmaz, "Estimating driving behavior by a smartphone," *Intelligent Vehicles Symposium*, 2012.

[6] C. Bo, X. Jian, T. Jung, J. Han, X.-Y. Li, X. Mao, and Y. Wang, "Detecting drivers smartphone usage via nonintrusively sensing driving dynamics," *IEEE Internet of Things Journal*, 2017.

[7] J. Yang, S. Sidhom, G. Chandrasekaran, T. Vu, H. Liu, N. Cecan, Y. Chen, M. Gruteser, and R. Martin, "Detecting driver phone use leveraging car speakers," in *Proceedings of the 17th Annual International Conference on Mobile Computing and Networking*, 2011, pp. 97–108.

[8] Y. Wang, J. Yang, H. Liu, Y. Chen, M. Gruteser, and R. P. Martin, "Sensing vehicle dynamics for determining driver phone use," *Proceeding of the 11th annual international conference on Mobile systems, applications, and services*, 2013.

[9] J. Yang, S. Sidhom, G. Chandrasekaran, T. Vu, H. Liu, N. Cecan, Y. Chen, M. Gruteser, and R. P. Martin, "Detecting driver phone use leveraging car speakers," *Proceedings of the 17th annual International Conference on Mobile computing and networking*, 2011.

[10] S. Gelmini, S. Strada, M. Tanelli, S. Savaresi, and V. Biase, "Analysis and development of a novel algorithm for the in-vehicle hand-usage of a smartphone," *Proceedings of the 2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2018.

[11] K. B. Ahmed, B. Goel, P. Bharti, S. Chellappan, and M. Bouhorma, "Leveraging smartphone sensors to detect distracted driving activities," *IEEE Transactions on Intelligent Transportation Systems*, 2018.

[12] B. P. Bogert, M. J. Healy, and J. W. Tukey, "The quefrency analysis of time series for echoes: cepstrum, pseudo-autocovariance, cross-cepstrum and saphe cracking," *Proceedings of the Symposium on Time Series Analysis*, 1963.

[13] O. Lauwers and B. De Moor, "A time series distance measure for efficient clustering of input/output signals by their underlying dynamics," *IEEE Control Systems Letters*, 2017.

[14] "One plus," http://oneplus.net/one.

[15] K. De Cock, *Principal angles in system theory, information theory and signal processing*. Ph.D. thesis, 2002.

[16] D. G. Childers, D. P. Skinner, and R. C. Kemerait, "The cepstrum: A guide to processing," *Proceedings of the IEEE*, 1977.

[17] R. J. Martin, "A metric for ARMA processes," *IEEE transactions on Signal Processing*, 2000.

[18] A. Eronen and A. Klapuri, "Musical instrument recognition using cepstral coefficients and temporal features," *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2000.

[19] C.-H. Lee, J.-L. Shih, K.-M. Yu, and H.-S. Lin, "Automatic music genre classification based on modulation spectral analysis of spectral and cepstral features," *IEEE Transactions on Multimedia*, 2009.

[20] K. Kalpakis, D. Gada, and V. Puttagunta, "Distance measures for effective clustering of arima time-series," 2001.

[21] J. Han, J. Pei, and M. Kamber, *Data mining: concepts and techniques*. Elsevier, 2011.