Texting and Driving Recognition leveraging the Front Camera of Smartphones

Federico Montori
University of Bologna
Bologna, Italia
federico.montori2@unibo.it

Marco Spallone
University of Bologna
Bologna, Italia
marco.spallone3@studio.unibo.it

Luca Bedogni
University of Modena and Reggio Emilia
Modena, Italia
luca.bedogni@unimore.it

***Abstract*—The recognition of the activity of texting while driving is an open problem in literature and it is crucial for the security within the scope of automotive. This can bring to life new insurance policies and increase the overall safety on the roads. Many works in literature leverage smartphone sensors for this purpose, however it is shown that these methods take a considerable amount of time to perform a recognition with sufficient confidence. In this paper we propose to leverage the smartphone front camera to perform an image classification and recognize whether the subject is seated in the driver position or in the passenger position. We first applied standalone Convolutional Neural Networks with poor results, then we focused on object detection-based algorithms to detect the presence and the position of discriminant objects (i.e. the security belts and the car window). We then applied the model over short videos by classifying frame by frame until reaching a satisfactory confidence. Results show that we are able to reach around 90% accuracy in only few seconds of the video, demonstrating the applicability of our method in the real world.**

***Index Terms*—image recognition, object detection, internet of things**

I. Introduction

Recognizing actions inside a car allows a computing system to reconfigure itself according to the different needs and status of the passenger. Specifically, it is possible to recognize statuses like driving fatigue, seats occupancy and driving inattention. By recognizing these behaviors a computing system can then act accordingly, for instance suggesting the driver to have some rest in case an excessive fatigue is recognized, or to avoid distracting elements for a safer driving. This field of research is named Context Aware Computing, in which a computing system anticipates possible scenarios according to the feedback from a number of sensors, and acts accordingly in order to improve the context itself. This involves (i) a first phase in which we need to recognize a set of behaviors, actions and statuses relevant to the scenario taken into account, (ii) a possible fusion of these information into an enhanced context, and (iii) a system which reacts according to the context by performing specific actions. In this paper we focus on the first stage of this process, more in detail we concentrate our efforts in the so-called Activity Recognition task, which refers to the possibility to classify behaviors performed by an individual or a group of users. More in detail, we study how to recognize a human being driving a car or being a passenger in it, by leveraging the front camera of an off-the-shelf smartphone. Enabling this recognition would then allow a Context Aware System to deny the use of a smartphone whilst driving, to improve the attention of drivers on roads. In fact the bad habit of *texting and driving* is commonly referred to as one of the major issues in modern driving scenarios, since drivers using the smartphone have less attention for the surroundings, leading to an increased number of accidents worldwide [1]. Using a smartphone whilst driving is forbidden in many countries, but it is reported that many individuals still do it regularly. This leads to alarming numbers, with more than 400,000 injuries related to distracted driving in 2012, which led to 3,328 people killed, and similar numbers in more recent years, despite the increase of automated safety measures employed by modern cars.

There have been several proposals to improve road safety by making drivers more conscious of the dangers of texting and driving, such as EverDrive[1] and DriverMode[2], which are applications that can be installed on the smartphone and block it in case the texting and driving action is recognized. All these applications rely on a voluntary installation by the user, hence they are rarely used, thus less effective. This requires solutions at the operating system level, which needs to be reliable and do not require external applications to work. In fact, the European Union mandated that starting from July 2022 all new cars must have a black box to record the vehicles parameters, hence a similar path could also be taken for smartphones used whilst driving. To achieve this, several solutions are possible, which may involve a different set of sensors and how those are placed. In this work, we leverage the front camera available in the majority of the smartphones to recognize the position of the individual using the smartphone. Any system can then easily leverage such classification to perform the needed actions in case the driver is using the smartphone, or simply ignore it if he/she is the passenger. We compare our system against other proposals from literature, and we show that our contribution achieves better and faster recognition of the driving individual, compared to similar solutions.

[1]EverDrive - Available Online: `https://play.google.com/store/apps/details?id=com.everquote.everdrive`

[2]DriverMode - Available Online: `https://play.google.com/store/apps/details?id=com.drivemode.android`

The rest of this paper is structured as follows: Section II describes related work from literature; Section III describes the scenario and the issues; Section IV describes the binary instantaneous classification while Section V takes into account object detection. Finally, Section VI extends our approach to continuous classification. Section VII concludes the paper and discusses future works on this topic.

II. Related works

Human Activity Recognition (HAR) is a set of techniques aimed at recognizing human activities, and it is a wide and actively researched topic. It is a key aspect of Context Aware Computing, given that it serves as the perception of the computing system towards a human behavior [2]. There have been a number of different proposals for HAR, which focus on different scenarios and behaviors of human life, such as Transportation Mode Detection (TMD) [3], which focuses on determining how users move, or home activities [4]. Apart from the specific scenario envisioned, we can broadly classify these proposals into inertial sensor based, vision based or a mix of the two techniques. Moreover, they could also be further classified upon using wearable devices or external ones, depending on where the sensing device is placed. When using inertial sensors, typically subjects have a wearable device that records their movements, which are eventually classified using Machine Learning or Deep Learning techniques. This is the case for instance of [5], in which the authors leverage inertial sensors and utilize them to classify human actions related to the transportation mode. Still for TMD it is also worth to cite [6], which uses GPS to recognize different vehicle speeds and classify them. Concerning the primary topic of this work, we note the work done by [7], which is focused on determining the attention of the driver. Specifically, they analyze the human head movements, and classify them into focused or non focused classes, to eventually determine the driver attitude. Other proposals include [8] and [9], which focus on the same task using different techniques. For a general overview of these techniques, we refer the interested reader to [10]. Closer to the work presented in this paper we mention [11], where the authors use external cameras to detect distracted driving, which happens for instance when drivers are drinking, looking at the back seats, or lower their head. In this case several photos of users are collected with a camera placed close to the rear view mirror in the center of the car, and used to train a Convolutional Neural Network (CNN). A similar approach is also studied in [12], where authors placed a camera in the car, which monitors the actions of the users, including reading a book or looking at their smartphone. This is also the approach undertaken in [13], where the study focuses more on CNN to classify non-driving activities.

In general, approaches based on inertial sensors struggle whenever the driver performs more complex actions, or simply purposely tricks the system, such as [14]. Here, the driver can simply leave the smartphone on the passenger seat, and the system would believe that the device is actually used by the passenger herself. This happens similarly to [8], since a significant number of curves, hence a longer recognition time, are needed in order to correctly classify the position of the smartphone. Moreover, all proposals based on intra-body devices suffer from users that can deliberately de-activate or make them unable to classify their behavior, by simply removing them from their body. Hence, mainly external sensing devices, such as cameras, are envisioned, as they monitor the environment without granting the user the ability to deactivate them. We note that these can be darkened, however this could be easily recognized and notified appropriately.

III. Scenario description

Texting and driving is certainly a serious problem, which affects also many new generations. Contrary to previous works [8] [14] we do not use inertial sensors to perform our classification, as they are prone to potential errors as already discussed. Instead, in this work we explore the possibility to use directly smartphones, as the texting activity is performed necessarily using them. Moreover, in contrast with external solutions, our solution does not require any additional hardware that needs to be installed in the car. Specifically, we leverage the front camera of the smartphone, which points directly at the user that is using the smartphone. Smartphone front cameras are generally wide enough in their field of view to have the user in the frame, along with other key distinctive elements which we leverage to determine the user position inside the car. Our primary task is hence to recognize whether users are on the left or on the right seat of the car. This inherently translates into knowing whether the user is actually driving the car or being a simple passenger.

At first, we explore the possibility of tackling this problem as a binary classification problem, with the two classes being "driver" and "passenger". We will discuss this in Section IV, where we will also explain why it does not achieve satisfactory performances. To improve the system, we will then transform the task into object detection, able to recognize the belt and the windows of the car. These two elements, although present both for the passenger and for the driver, are in different positions within the frame, also the belt is bent in a different angle. We will show how this improves the overall accuracy of the system. Finally, we will present our final system, which is composed by object detection and correlating subsequent images. More in detail, we will reduce the uncertainty in the classification task shown in Section IV by classifying objects in subsequent images. Numerical results confirm that this technique provide superior results both in terms of accuracy as well as recall, and is also much faster compared to existing works such as [8] and [14].

IV. Instantaneous classification

In this section we outline the steps taken towards solving the image classification problem in a traditional way. This means leveraging well-known and established techniques for accomplishing a binary classification task over images.

The first step towards this goal is creating a consistent dataset for the experiments. In our case, the *Instantaneous*

Fig. 1: Example of an image belonging to the *Instantaneous* dataset, with the security belt and the car window visible. In this case, the person is a driver. The face was anonymized for privacy.

dataset, used within this section, is solely composed by standalone images, very much like the example image reported in Figure 1. More in detail, the dataset was created by one of the authors by taking 50 photos of himself while sitting in one of the front seats of the car. In 50% of the cases the subject is sitting on the left seat, in the remainder in the right seat. Note that we did not consider a subject sitting in the back seat for the purpose of the present paper, as it would make the problem very complex and a solution would require additional sensory data in the loop. The photos were taken from different angles, both at day time and night time; the subject in the dataset is the same person, wearing different clothes and always wearing the security belt, which is visible in every picture. We then performed a data augmentation step that increased the number of images in the dataset from 50 to 4700 and homogenized their size to 360×360. We used the `ImageDataGeneration` function, setting its parameters as follows: rotation range $= 45$, shear range $= 0.2$, zoom range $= 0.4$, brightness range $= (0.1, 0.5)$, horizontal flip $= False$, fill mode $=$ "$nearest$", cval $= 125$. All images were manually labeled before augmentation as belonging to one of "driver" or "passenger" – the horizontal flip was set to $False$ as it would invert also the image class. For the sake of simplicity, we hereafter consider the driver to be the one on the left seat, therefore for countries where people drive on the right. The

Model	Accuracy
Custom CNN	56.7%
ResNet50	52.3%
COCOv6n	68.7%

TABLE I: Accuracy of instantaneous classification using different models and approaches.

dataset was then divided into training and validation sets, with a percentage of 80% and 20%, respectively, and split into batches of 18 images, in order to be able to train a model safely with Google Colab[3] without incurring into RAM overflows.

A. Custom Convolutional Neural Network

On top of the created *Instantaneous* dataset, we then proceeded to create a Convolutional Neural Network (CNN) for the classification task. In this first experiment the CNN is fairly simple and standard, with a first level of rescaling, then three convolutional layers, each of them using a 3×3 kernel and followed by its respective *MaxPooling* layer. Afterwards, a single flatten layer was applied, which was then given in input to a dense layer with a ReLu activation function and a second dense layer that outputs two classes. The model was written using Tensorflow via the Keras library and run through an Adam optimizer instead of a classic gradient descent [15]. The CNN is evaluated along 15 epochs using the *accuracy* as a driving parameter.

B. Transfer Learning

Using the *Instantaneous* dataset as a basis, we also explored the utilization of Transfer Learning techniques for our problem. This basically means exploiting existing models already trained on extensive generalist datasets [16]. An example of such datasets is ImageNet, one of the most popular and complete, which comprises millions of images and more than 1000 classes. The idea is then to leverage prior knowledge instead of training a model from scratch, in order to extract useful information for a more specific task, like ours. Technically, a Transfer Learning process consists in adapting the preexisting model to our problem, by adding a few layers that fit our dimensional needs. In particular, we chose to use ResNet50, trained over ImageNet, by flattening its output and adding the same dense layers that we added in our custom CNN in Section IV-A. We then performed an similar learning process, in order to obtain coherent results.

C. Preliminary Evaluation

We performed extensive performance tests over the validation set, however the average accuracy obtained through the presented models was a quite disappointing result. The percentages are shown in Table I, from which it is evident that, for a binary classification task, the plain application of a Neural Network is almost equivalent to a stab in the dark. This result might have been driven by a number of

[3] `https://colab.research.google.com/`

factors, for instance the small size of the dataset and the differences between images, given by the day/night contrast and the different clothes. We thus repeated the experiments by only keeping the daytime pictures and only one set of clothes, however this did not improve significantly the accuracy, which led us to completely change our approach. The intuition, explained in more detail in the subsequent section, is that there are very few elements in the photos that characterize the two classes, while the others are just noise. Since it is difficult to instruct generic models to concentrate only on few details, we thought that an object detection step could greatly improve the performance of the system.

V. Object Detection

By using the same *Instantaneous* dataset, we applied a completely different approach based on *object detection*. The rationale behind this choice is that we realized generic methods are not working, because they may take into account many aspects that are not directly correlated with the classes. In our case, instead, we observe that there are very few distinctive elements in the photos that can discriminate the class with certainty: the security belt and the car window. In particular, the direction of the security belt is a very determinant feature, as it is opposite for the two classes. Similarly, the car window on the right of the picture suggests that the subject is seated on the left and vice versa. If we restrict our classification task by finding those objects, we expect a much higher performance. On top of these premises we proceeded once again to train the object detection model using a part of the training set. We identified namely four different objects: the driver's window (DW), the passenger's window (PW), the driver's security belt (DB) and the passenger's security belt (PB). We then used the tool *LabelImg*[4] to locate the objects manually in the training set. In practice, through such a tool we identify the elements by manually drawing bounding boxes around them and giving them a label – as shown in Figure 2 – this generates an XML file with the coordinates of the bounding boxes and their label. The training phase has been performed through the tool Roboflow[5], which is commonly used for this type of tasks. The user uploads images together with their XML label file and sets the percentage of training and test sets. Subsequently the pre-trained model should be selected, in our case we chose COCOv6n, based on the COCO (Common Object Context) dataset [17]. Roboflow allows also to test the model interactively, by uploading a new image, where objects are recognized graphically. We then evaluated the model over the test set with the following criterion:

- If the model recognizes more elements related to the driver (DB and DW) than the ones related to the passenger (PB and PW) in the test example, then its predicted class will be "driver".
- If the model recognizes more elements related to the passenger than the ones related to the driver in the test example, then its predicted class will be "passenger".

[4]https://github.com/heartexlabs/labelImg
[5]https://roboflow.com/

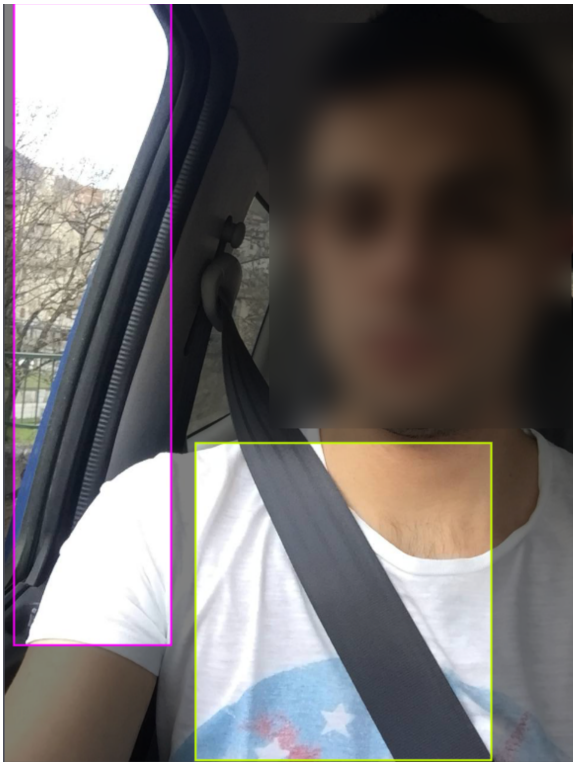

Fig. 2: Example of a training image after its labeling through *LabelImg*. The purple bounding box has been labeled with PW, whereas the yellow one has been labeled with PB. The face was anonymized for privacy.

- If the model recognizes an equal number of elements related to the driver and the passenger in a test example, then its associated class will be "uncertain" and it is treated differently depending on the type of classification.

We then performed extensive tests much like the ones taken in Section IV-C. Since in this case the classification is standalone and stateless, we treated the text examples that are "uncertain" just like a coin flip (i.e. we assign randomly a predicted class between driver and passenger). The classification yields an average result of 68.7% – also reported in Table I. This validates our hypothesis that focusing on specific discriminating elements yields a much better result. Furthermore, we also observed that a fair number of test examples were uncertain – around 47.5% – and the actual false positives were very few – circa 7.5%. This suggested us to focus on the uncertain examples in order to classify them correctly. Our proposed solution is to leverage images in sequence belonging to the same drive until one of them gets a correct classification. This is explained in detail in the next section.

VI. Continuous Classification

Standalone or – as we called it – instantaneous classification through object detection suffers from the presence of many uncertain predictions. This observation led us to consider once again a real scenario: if the smartphone of a person in a car is able to take a picture of him/her, then it is certainly able

to take multiple ones in a sequence, related to the same car trip. We obviously assume that the subject does not change seat within the – reasonably short – time span in which such photos are taken. This opens up new possibilities as ideally we might consider to classify a set of photos instead of one and apply a simple ensemble policy, like majority voting. This would considerably limit the issue of uncertain samples, as, the bigger the set of pictures, the more likely is that a sufficient number of them are not uncertain.

In order to pursue this path, we need to consider a different dataset though, as the pictures in the *Instantaneous* dataset are not shot in sequence. For this reason, we created the *Continuous* dataset, used for performance evaluation within this section. The dataset was created as follows: one of the authors recorded 8 videos (half of them as a driver, half of them as a passenger) each one lasting 10 seconds, reproducing natural motion of the head and the wrist. Each of the video was then divided into frames with a rate of 15fps, resulting therefore in 150 frames each. This way we obtained a dataset of 1200 photos, each of them labeled with a class and organized in 8 sequences. In order to be coherent with the results shown in Section V, we used the same model without retraining it, nor adding more labels. A standalone test, analogous to the one in Section V by using the entire *Continuous* dataset as a test set, yields an accuracy of 80%.

In order to implement the continuous classification, we proceeded to group the video frames – *aka* the images – in time windows of length l using the sliding window approach. This means that a video with L frames will be divided into $L - l + 1$ windows. Clearly each window belongs to a single class and can be classified as a single instance. In this case, we follow the same approach as in Section V, however we sum all objects recognized in each of the photos in the window. This way, the longer the time window is, the less probable is to end up with an uncertain classification. Figure 3 shows the

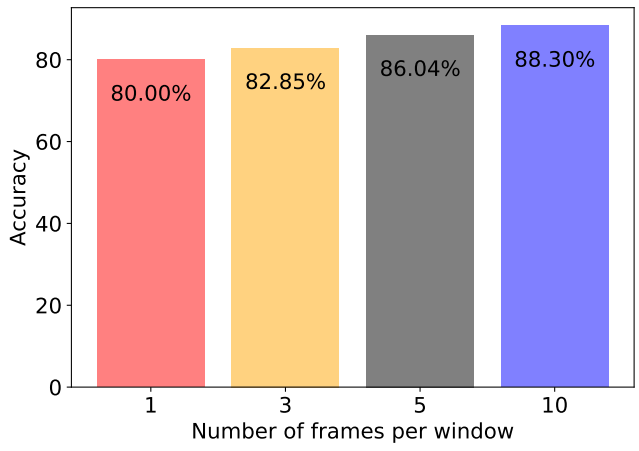

Fig. 3: Bar chart showing the accuracy of the object detection model over the *Continuous* dataset for different window sizes – i.e. different values of l.

accuracy of the trained model for different values of l, namely 3, 5, and 10. We also show the previously mentioned result for $l = 1$, which corresponds to the standalone classification. The plot shows a monotone increase of the accuracy as l grows, which is expected for two reasons: first, the uncertain frames that end up in a window with at least a true positive lose their uncertainty, second, the wrong classifications, as their number is tiny, might be easily dominated by correct classifications within the same window, as they are likely to be more.

This encouraging result led us to think about how to apply such a system in a real scenario. If we take into account the scenario described in Section III, we have that the smartphone can record a number of pictures of the subject. The question is: how many pictures must we consider in order for the system to be certain about its classification result? In other words, how long should the smartphone record images before reaching confidently a verdict? To answer these questions we performed an additional evaluation step. Taking as an example a single video, then we considered the first γ windows of length l. We then perform the classification on each of them, obtaining γ predictions. We then introduce the *confidence* value k as the maximum number of predictions of the same class, expressed as a percentage over the total number of predictions. For instance, if $\gamma = 10$ and the classification step predicts 7 windows as driver, 2 windows as passenger and 1 window as uncertain, then $k = 70\%$. If k is satisfactory, we consider the verdict as final, otherwise we add a subsequent time window, considering then $\gamma + 1$ windows and recalculating k. We consider k to be satisfactory if $k \geq K$, where K is a fixed predetermined confidence threshold. We assume that the application manufacturer uniquely sets K once the application is in production, after a thorough evaluation step. We then performed the above evaluation over the videos, halting the execution as the threshold K was reached. Figure 4 shows the evaluation performed for five different values of K and seven different values of l, by setting $\gamma = 10$. The y-axis shows the time takes until K is reached. Note that this is not the actual execution time of the algorithm; the time was calculated statically on top of the number of frames that were taken into account; let us call it "video time". Since the videos were recorded at 15fps, each considered frame accounts for roughly 0.067s. First of all we notice that, as expected, for a fixed value of l, the video time increases as K increases, as the confidence demanded is higher, then more classification steps need to be performed in order to be K-sure of a certain verdict. What is really interesting though, is that, for a fixed value of K, the video time does not always have a monotone behavior as l increases. In particular for really high values of K – i.e. $K \geq 90\%$ – the video time shows a local minimum, which seems to correspond to an optimal value of l, given the problem and its requirements. In this case, for instance, we can say that for the problem presented in this paper and a required confidence level of 99%, the optimal value of l among the ones considered is $l = 15$; bigger time windows are probably a waste of resources. Another takeaway message is that very high levels of confidence are obtained with very

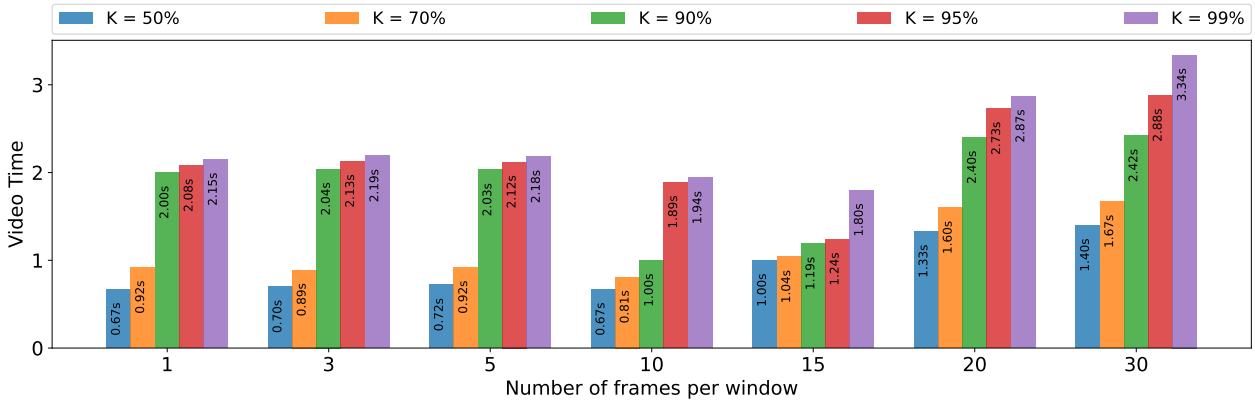

Fig. 4: Bar chart showing the video time taken to reach a confidence threshold K for different values of l and with $\gamma = 10$.

short videos (1-2 seconds) making this approach very solid and applicable to a real scenario. Finally, for the limited amounts of videos considered in this paper, we obtained 100% accuracy for any value of K, which further strengthens the confidence in our proposed method and suggests that, even for K small, results can be satisfactory. Future works aim at evaluating it over larger and all-encompassing datasets.

VII. Conclusion

In this paper we have provided an image classification solution to solve the problem of texting and driving. Ideally, an application hosting this system would stop the driver of a vehicle from using the smartphone if he or she is driving. In our case we classified the position of a subject seated in a car into "driver" or "passenger" by means of the smartphone front camera. We first approached the problem with classic algorithms, using CNNs to classify one image at a time (instantaneous classification). The poor results led us to approach the problem from an different angle, therefore we first shifted to an object detection-based approach, by identifying the direction of the security belt and the position of the car window. Then, we applied a continuous classification approach, by predicting the class of video frames in sequence, until the algorithm reaches a certain confidence. Results shown the applicability of the algorithm in a real world scenario.

References

[1] National Highway Traffic Safety Administration (NHTSA), "Report on distracted driving, available online: https://www.nhtsa.gov/risky-driving/distracted-driving."

[2] C. Randell and H. Muller, "Context awareness by analysing accelerometer data," in *Digest of Papers. Fourth International Symposium on Wearable Computers*, Oct 2000, pp. 175–176.

[3] L. Bedogni, M. Di Felice, and L. Bononi, "Context-aware android applications through transportation mode detection techniques," *Wireless Communications and Mobile Computing*, vol. 16, no. 16, pp. 2523–2541, 2016, wcm.2702.

[4] V. Bianchi, M. Bassoli, G. Lombardo, P. Fornacciari, M. Mordonini, and I. De Munari, "Iot wearable sensor and deep learning: An integrated approach for personalized human activity recognition in a smart home environment," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8553–8562, 2019.

[5] C. Carpineti, V. Lomonaco, L. Bedogni, M. D. Felice, and L. Bononi, "Custom dual transportation mode detection by smartphone devices exploiting sensor diversity," in *2018 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, 2018, pp. 367–372.

[6] S. Reddy, M. Mun, J. Burke, D. Estrin, M. Hansen, and M. Srivastava, "Using mobile phones to determine transportation modes," *ACM Trans. Sen. Netw.*, vol. 6, no. 2, pp. 13:1–13:27, Mar. 2010.

[7] M. García-García, A. Caplier, and M. Rombaut, "Driver Head Movements While Using a Smartphone in a Naturalistic Context," in *6th International Symposium on Naturalistic Driving Research*, vol. 8, The Hague, Netherlands, Jun. 2017.

[8] Y. Wang, Y. J. Chen, J. Yang, M. Gruteser, R. P. Martin, H. Liu, L. Liu, and C. Karatas, "Determining driver phone use by exploiting smartphone integrated sensors," *IEEE Transactions on Mobile Computing*, vol. 15, no. 8, pp. 1965–1981, Aug 2016.

[9] X. Liu, J. Cao, S. Tang, Z. He, and J. Wen, "Drive now, text later: Nonintrusive texting-while-driving detection using smartphones," *IEEE Transactions on Mobile Computing*, vol. 16, no. 1, pp. 73–86, Jan 2017.

[10] M. Cismaru and K. Nimegeers, "Keep your eyes up, don't text and drive: a review of anti-texting while driving campaigns' recommendations," *International Review on Public and Nonprofit Marketing*, vol. 14, no. 1, pp. 113–135, Mar 2017.

[11] S. Masood, A. Rai, A. Aggarwal, M. Doja, and M. Ahmad, "Detecting distraction of drivers using convolutional neural network," *Pattern Recognition Letters*, 2018.

[12] L. Yang, K. Dong, Y. Ding, J. Brighton, Z. Zhan, and Y. Zhao, "Recognition of visual-related non-driving activities using a dual-camera monitoring system," *Pattern Recognition*, vol. 116, p. 107955, aug 2021.

[13] J. M. Celaya-Padilla, C. E. Galván-Tejada, J. S. A. Lozano-Aguilar, L. A. Zanella-Calzada, H. Luna-García, J. I. Galván-Tejada, N. K. Gamboa-Rosales, A. Velez Rodriguez, and H. Gamboa-Rosales, ""Texting & Driving" Detection Using Deep Convolutional Neural Networks," *Applied Sciences*, vol. 9, no. 15, 2019.

[14] L. Bedogni, O. Bujor, and M. Levorato, "Texting and driving recognition exploiting subsequent turns leveraging smartphone sensors," in *2019 IEEE 20th International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM)*, 2019, pp. 1–9.

[15] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[16] L. Torrey and J. Shavlik, "Transfer learning," in *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*. IGI global, 2010, pp. 242–264.

[17] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*. Springer, 2014, pp. 740–755.