# Real-time lane classification and accident detection for safer micromobility

**Morros, Josep Ramon[1]*, Broquetas, Antoni[2], Mateo, Alex[3], Puig, Jordi[4], Davins, Manel[5]**

[1,2,3,4]Universitat Politècnica de Catalunya (UPC). Signal Theory and Communications Department
[5]CARNET Barcelona

E-mail[1] *corresponding author: ramon.morros@upc.edu
E-mail[2]: antoni.broquetas@upc.edu
E-mail[3]: antoni.broquetas@upc.edu
E-mail[4]: antoni.broquetas@upc.edu
E-mail[5]: manel.davins@carnetbarcelona.com

**Abstract**

The lack of knowledge of the micromobility regulations by e-scooter users is an important factor behind some of the accidents involving these vehicles. We present two modules that can increase the safety of the users and pedestrians: First, a computer vision model that analyses the video feed captured with a smartphone attached to the e-scooter, and predicts in real-time the type of lane in which the user is riding. This knowledge is used by an application which combines this information with GSNN location information and a database of mobility regulations, and informs the user when he/she is not complying with these regulations. Second, an accident detection system, using the smartphone accelerometer, that detects if there is a fall during the riding, so that the app can contact the authorities to determine the appropriate response. The experimental results show excellent results for both modules.

***Keywords:*** *Micromobility, accident detection, lane classification, RideSafeUM, computer vision, deep learning.*

## *1. Introduction*

In recent years, there has been a strong increase in the use of e-scooters. These vehicles allow a fast and convenient mobility, they are low cost and almost pollution free. But the reckless use of these vehicles, such as not properly using bike lanes, or riding on footpaths, may result in accidents involving the riders and the pedestrians nearby. One of the main reasons for accidents is that a lot of the users do not know local micromobility regulations.

In this work, we present two of the technical solutions developed as part of the RideSafeUM project (https://ridesafeum.com), co-funded by the EIT Urban Mobility. The first solution is a computer vision model that can determine the type of lane the user is riding on from the recordings provided by a camera placed in front of the vehicle (scooter or bicycle). The type of lane (sidewalk, bike lane or road) is forwarded in real-time to the RideSafeUM app along with the GNSS location of the vehicle. With this information, the app can give warnings when the user is not complying with the regulation. The second solution is an accident detection system that can determine if an accident has occurred, permitting the app to inform the authorities and emergency services automatically. These solutions must work in real-time in the rider's mobile phone, which is installed on the handlebar with the main camera pointing in the direction of the ride.

For the lane recognition computer vision module, an image classifier has been trained to recognize three different lane types: sidewalk, bike lane and road. A computationally efficient convolutional neural network is used in order to allow real-time operation in mid-level mobile phones. We have trained the model with a database of 1000 videos of 45' captured in Barcelona. The final algorithm averages the results of several frames to give a classification each second. This information is forwarded to the app (not explained here), which can inform the user of any violation of traffic rules. We achieve a classification precision of 0.94 with a recall of 0.85, which has resulted in a satisfying operation during last year's user's tests.

For the accident detection module, a method based on fall detection when the vehicle is riding is proposed. The accident fall down can be detected by observing the direction changes of the gravity vector with respect to vehicle axes from standing when circulating to a fall down position. After transforming smartphone measurements into vehicle 3D accelerations, the fall is detected when the gravity vector is closely parallel to the vehicle platform plane.

The accident detection algorithm performs a binary integration of single detections, resulting in improved detection and false alarm probabilities. In this way, an accident is declared if a riding vehicle provides at least 50 fall detections in a group of 100 successive acceleration measurements. With this approach the experimental record of the detector is excellent with no reported detection errors in the performed tests.

## *1.1 Related work*

**Efficient image classification models**

Classification is a predictive task where a given model receives an input and assigns a category or class to it. In machine learning, the model learns how to perform the prediction by adjusting its parameters from a set of input examples, called training data. The model is expected to generalize the predictions to examples not present in the training dataset. A very successful family of classification models are based on Convolutional Neural Networks (Goodfellow, Bengio and Courville, 2016). This type of networks allow to extract discriminative features from the images and to use these features to perform the classification task. However, in many cases, CNNs are very computationally intensive. For this reason, a lot of efforts have been devoted to improve the efficiency of this type of networks, specifically in terms of inference time, memory usage and energy consumption, while maintaining the accuracy of the original models. The result has been the creation of a set of new architectures called Efficient CNNs.

For instance, SqueezeNet (Iandola *et al*, 2016) employs 1x1 convolutions with squeeze and expand modules to reduce the number of parameters. In (Howard, A.G., *et al.,* 2017), a class of network architectures (Mobilenet v1) is proposed using depth-wise separable convolutions, which result in a reduced number of model parameters and operations. Additionally, the resolution of the input image and the number of network channels can be modified. The Mobilenet v2 (Sandler, Howard, Zhmoginov and Chen, 2018) adds to this architecture a novel layer module: the inverted residual with linear bottleneck. This allows to significantly reduce the memory footprint needed during inference. ShuffleNet (Zhang, Zhou, Lin and Sun, 2018) further reduces the number of operations by using group convolutions and channel shuffle operations. ShuffleNet v2 (Ma, Zhang, Zheng and Sun, 2018) further optimizes the network architecture by introducing a channel split operator. MobileNetV3 (Howard *et al*., 2019) is tuned to mobile phone CPUs through a combination of hardware aware network architecture search (NAS) complemented by the NetAdapt algorithm and then subsequently improved through novel architecture advances. The authors of EfficientNet (Tan and Le, 2019) propose a scaling of CNNs to obtain better results by increasing the efficiency of the models.

**Accident detection using accelerometers**

Accident detection in the automotive industry is a widely studied field. The research and products concentrate mainly on cars, generally using several accelerometers in combination with other sensors. For instance, these algorithms are used to deploy the airbags in case of an accident. Some works also focus on accident detection by using smartphones (Aloul, Zualkernan, Abu-Salma, Al-Ali and Al-Merri, 2014; Hirawat and Bhargava, 2015; Lahn, Peter, and Braun, 2015; White, Thompson, Turner, Dougherty and Schmidt, 2011). There are commercial products like Google Car Crash detection (Snyder, Ferguson and Irey, 2015) which are on the market nowadays. Here, the main difficulty is that the smartphone may not be firmly attached to the car and user movements of dropping the phone may cause false alarms. Threshold-based techniques using accelerometer data have been previously proposed to detect and report accident in motorcycles (Watthanawisuth, Lomas and Tuantranont, 2012) which is an application similar to ours. Finally, in (Kau and Chen, 2015), accelerometers have been used to detect generic falls using a cascade classifier.

### *1.2 Paper structure*

This paper is structured as follows: this introduction provides the overall goals of the work, as well as a literature review of efficient classification models. Section 2 provides a detailed explanation of the technical solution. The hardware setup is presented. Then the dataset used to train the computer vision models is described. Lastly, the computer vision model and the accident detection module are fully explained. Finally, in section 3 overall conclusions are presented and future research lines are provided.

## *2. Technical solution*

### *2.1 Hardware setup*

The setup used in the project is composed of a mobile phone attached to the handlebar of an e-scooter (see Figure 1), with the camera facing forward. The video frames recorded by the camera are processed one by one by a classification network, which provides the probabilities to belong to each of the three predefined classes.

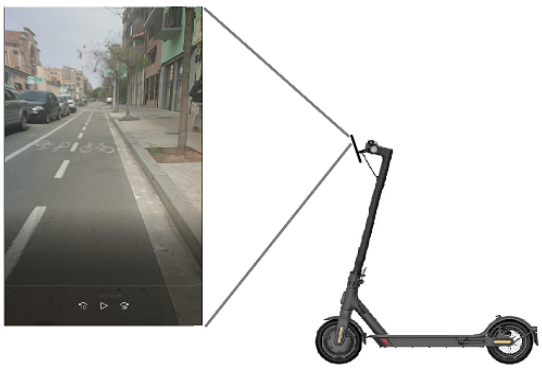

Figure 1: Hardware setup

### *2.2 Video dataset*

The database used in this work consists of a set of 858 videos recorded in the city of Barcelona in 2022, amounting to a total of 8 hours. They have been recorded using the setup described in Section 2.1. Videos were captured at a resolution of 1920 x 1080 at 60 fps using a Xiaomi Redmi Note10 Pro and a Pixel 3 XL smartphones. The videos were captured in different days, locations, weather conditions and types of roads to maximize the variability and to ensure that the experimental validation reflects the real

riding conditions as much as possible. The videos belong to the 3 possible classes: sidewalk, road and bike lane. Examples of each type of lane can be seen in Figure 2.

a) Bike lane b) Road c) Sidewalk

Figure 2: Types of lanes.

The recordings are no longer than 40 seconds and each one contains only one type of road, to facilitate the manual labelling of the videos.

Table 1 shows the distribution of the videos between the three classes. It can be seen that the dataset is mostly balanced between the three classes.

Table 1: Class distribution of the dataset.

| **Class** | **Bike lane** | **Road** | **Sidewalk** | **Total** |
|---|---|---|---|---|
| Number of videos | 318 | 268 | 272 | 858 |
| Total time | 2h 56’ 28’’ | 02h 20’ 12’’ | 02h 52’ 31’’ | 8h 9’ 11s |

Together with the videos, we recorded the GPS coordinates of the routes. This allowed to map the locations of the recordings (see Figure 3) to make sure that the dataset is representative of the different streets, and also to avoid recording the same streets more than once. There are no recordings on the Barcelona old town because this is an area mostly restricted to e-scooters.

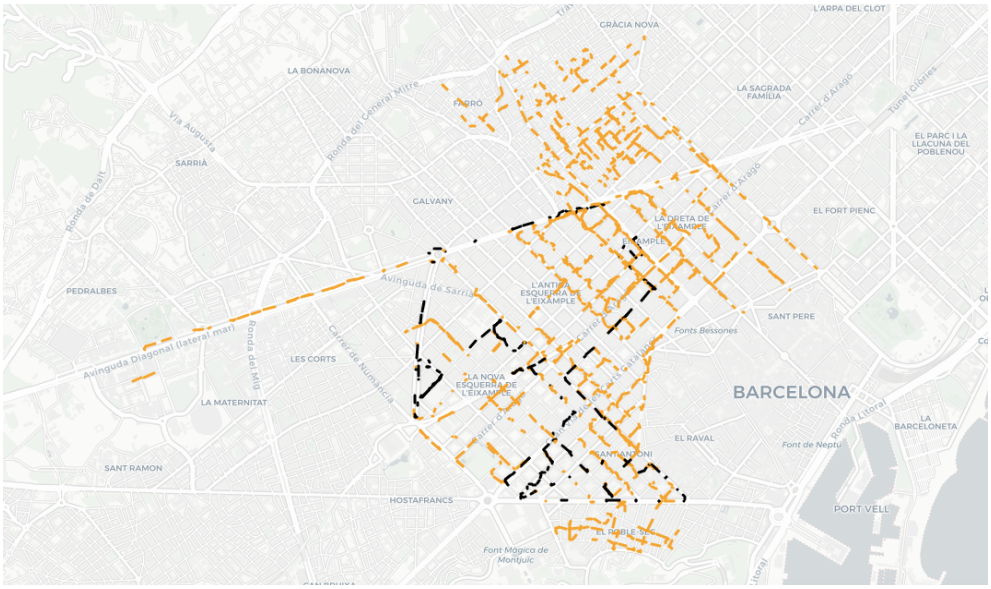


Figure 3: Map with the recorded streets of Barcelona. Orange traces represent videos recorded during the day while black ones were recorded at night.

### *2.3 Lane classification using Deep Learning*

For the lane classification, several efficient networks were tested (Mateo, 2022; Puig, 2022) and the use of a ShuffleNet v2 1x was decided. This version has 1.3 M parameters and allows processing at least 6 fps at the selected resolution in a mid-range smartphone, which proved sufficient for our application.

We started with a base model pretrained on ImageNet (Deng *et al.*, 2009) and fine-tuned it using our dataset. For the training, each frame of the sequence was downsampled to 256x456 pixels. We used data augmentation (horizontal flip, small rotations and Gaussian blur) to prevent overfitting. The dataset was split into 579 videos for training, 194 for validation and 195 for testing, corresponding to a 60%/20%/20% split. The model was fine-tuned for 10 epochs with the following hyperparameters:

Learning rate: 1e-4, optimizer: Adam, batch size:128, weight decay: 1e-3, 10 epochs.

We used a random sampler weighted by the inverse frequency of the different classes to balance the dataset during the training step.

As we used a SoftMax classifier, the output of the model is the class probabilities for each processed frame. The class with larger probability is selected as the final decision for this frame.

To give meaningful information to the e-scooter user, outputting a result per each processed frame is not practical, as the results could be noisy and change too fast for a good understanding. For this reason, an aggregation of the single frame results is performed over a temporal segment of **S** seconds. This is performed by averaging all the frames processed in a second. To simulate the conditions in a smartphone, which cannot process frames at 30 fps, we selected to perform temporal subsampling, processing just **N** frames per second. Thus, each **N** frames, the probabilities of the different classes are averaged, and the class with resulting larger probabilities is decided for this period. After extensive experiments (Mateo, 2022) we selected to use **S**=1s and **N**=6 fps.

As the goal of the application is to inform the user not of the type of lane, but of possible infractions to the traffic rules, we decided that avoiding false alarms was more important than to inform of all the infractions. This is because the use of the application is optional, and if the user receives several false alarms, he/she can decide to not use the app anymore.

To avoid giving false warnings, we do not issue a decision in the cases where the classification of a temporal segment is not clear. This can be achieved using two different mechanisms:

a) Setting a threshold in the confidence score for the segment. A confidence score for the segment can be computed by averaging the probabilities of the individual frames. Then, segments with probabilities lower than the threshold are classified as 'Unknown', and the app does not issue any warnings in this case.
b) Imposing a temporal coherence condition. In a normal ride, changes of lane are not frequent, and there are periods of many one second segments between changes of lane. In this case, the decision for the current time period should be the same as the decision for the $k$ previous segments. We can leverage this temporal coherence condition by comparing the decision in the current segment to the decision of the $k$-$1$ previous segments. If the decisions in this sliding window of $k$ segments are unanimous, this class is output for the current segment. Otherwise, the segment is classified as 'Unknown' and no warnings are issued by the app. We have selected a value of $k = 3$ (Mateo, 2022).

Of course, the two mechanisms can be combined to further remove the false alarms. When using the temporal coherence method, there is a delay of $k$ seconds to give the first result. Afterwards, there is no delay, but when there is a change of type of lane, the result of the prediction is 'Unknown' for at least $k$ frames. This is hardly noticed in the final application with the selected values (3 seconds).

The metrics used to evaluate the model are the multiclass, micro-averaged Precision and Recall (Pedregosa et al., 2011). These metrics are based on the concepts of True Positives (TP), False Positives (FP), False Negatives (FN) and True Negatives (TN). As we are in a multi-class scenario, the TPs, FPs and FNs are computed for each class. A TP for a class $i$ is when, for a segment of class $i$, the prediction given by the classifier is also class $i$. Otherwise, we have a FN for class $i$. A FP for class $i$ is when a segment of any other class different than $i$ is classified as class $i$. With these definitions, the precision for class $i$ can be seen as the fraction of TPs among all the positive's recalled, and the recall, as the fraction of TPs among all the correct. Note that segments classified as 'Unknown' do not count as FP, but do count as FN.

The final precision and recall can be computed using Equation 1:

$$\boldsymbol{P} = \frac{\sum_i \boldsymbol{TP_i}}{\sum_i \boldsymbol{TP_i} + \sum_i \boldsymbol{FP_i}}, \quad \boldsymbol{R} = \frac{\sum_i \boldsymbol{TP_i}}{\sum_i \boldsymbol{TP_i} + \sum_i \boldsymbol{FN_i}} \tag{1}$$

The prediction results of the trained classifier are shown in Table 2. The first row shows the precision and recall metric accumulated for a period of one second. Rows two, three, and four show the results by filtering the results using the probability threshold, the window unanimity and the combination of both, respectively.

Table 2: Results

| | Validation set | | Test set | |
|---|---|---|---|---|
| | **Precision** | **Recall** | **Precision** | **Recall** |
| **Segment** | 0.904 | 0.904 | 0.905 | 0.905 |
| **Segment, probability threshold** | 0.942 | 0.850 | 0.911 | 0.900 |
| **Segment, window unanimity** | 0.955 | 0.850 | **0.958** | **0.850** |
| **Segment, prob. & window** | 0.969 | 0.800 | 0.958 | 0.850 |

As we can see in Table 2, the two filtering methods increase the precision and decrease the recall. To compute the optimal threshold, we fixed a minimum recall of 0.85 in the validation set and selected the threshold that maximizes the precision. The value of 0.85 was decided 'ad-hoc', so that a significant percentage of the possible regulations' violations could be detected. The best results are when window unanimity is used, obtaining a precision of 0.958.

Note that not all classification errors result in a false warning to the user. For instance, in Barcelona e-scooters are not allowed to ride on the sidewalk and should always ride on bike-lanes or roads with a speed limit of 30 km/h. If the e-scooter is riding on a valid road and the system predicts incorrectly a bike-lane, no warning is issued. This means that the number of false warnings is really low.

Tables 3-6 show the test-set confusion matrices (CM) for the four experiments at the segment level. Rows show the ground-truth, and columns show the predictions. We can see that most confusions are between Bike and road classes, as the pavement is very similar in both cases. The most critical errors are those in the **S** column (marked in red in the tables), as predicting sidewalk when the user drives on a bike lane or a road will issue a warning (in Barcelona it is not allowed to ride an e-scooter in sidewalks). As we can see, when using the probability and window unanimity filtering, this type of errors is very unlikely.

Table 3: CM without filtering

| | **B** | **R** | **S** | **Unk.** |
|---|---|---|---|---|
| **Bike** | 1719 | 107 | 54 | 0 |
| **Road** | 165 | 1286 | 153 | 0 |
| **Sidewalk** | 17 | 0 | 1708 | 0 |

Table 4: CM with prob. filtering

| | **B** | **R** | **S** | **Unk.** |
|---|---|---|---|---|
| **Bike** | 1519 | 41 | 26 | 184 |
| **Road** | 86 | 1047 | 80 | 290 |
| **Sidewalk** | 8 | 0 | 1602 | 19 |

Table 5: CM with window filtering

| | **B** | **R** | **S** | **Unk.** |
|---|---|---|---|---|
| **Bike** | 1506 | 43 | 16 | 205 |
| **Road** | 68 | 1037 | 49 | 349 |
| **Sidewalk** | 8 | 0 | 1608 | 13 |

Table 6: CM with prob. & window filtering

| | **B** | **R** | **S** | **Unk.** |
|---|---|---|---|---|
| **Bike** | 1373 | 15 | 7 | 375 |
| **Road** | 31 | 861 | 23 | 588 |
| **Sidewalk** | 2 | 0 | 1584 | 43 |

### *2.4 Accident detection based on smartphone accelerometers*

Since present smartphones integrate 3D accelerometers, the first approach in the design of an accident detection system for micromobility was the detection of collisions from smartphone measured accelerations. However, after preliminary analysis and experimentation it was found that smartphone MEMS accelerometers devices have very limited full-scale acceleration specifications, often +/- 16g, +/- 8g or even lower depending on smartphone manufacturer settings. As a consequence, normal driving acceleration peaks often reach the accelerometers full-scale range values, compromising the reliable detection of collisions in accident detection applications. This is shown in Fig. 4 corresponding to a scooter collision occurring in Time ~ 128 s preceded by road roughness induced accelerations often reaching +/- 8g full scale values. Fortunately, the experimental scouter accident tests revealed a reliable accident detection alternative based on fall detection when the vehicle is riding. In 2-wheel vehicles, the accident fall down can be detected by observing the direction changes of the gravity vector with respect to vehicle axes from standing when circulating to a fall down position.

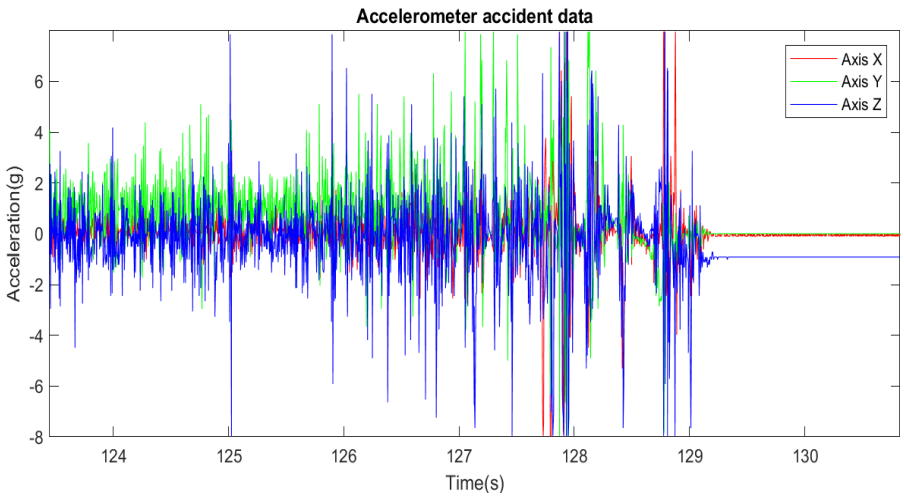


Figure 4: Pre- and post-collision accelerations of a scooter accident occurred at Time ~ 128 s. Note that with an accelerometer with typical full-scale of 8 g the collision is hardly distinguishable from normal riding vibration. However, the post-accident fall from Time = 129 s is easily detected

#### *A. Smartphone Tilt Calibration*

The fall detection requires adopting a vehicle 3D acceleration frame which is not parallel to the smartphone acceleration reference frame due to installation tilt angle α as shown in Fig.5

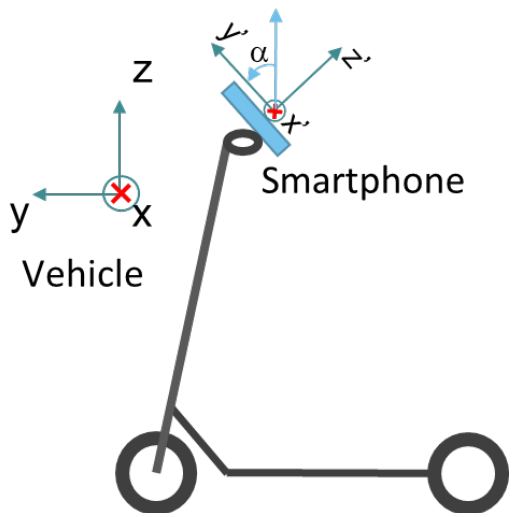


Figure 5: Definition of vehicle acceleration axes x,y,z and android smartphone accelerometer measurement frame x‘, y‘, z‘. Note the tilt installation angle α that must be known or calibrated to transform smartphone into vehicle acceleration vectors.

If the installation tilt angle α is known the vehicle Ax, Ay, Az accelerations can be easily computed from the trigonometric projections of smartphone accelerations Ax’, Ay’, Az’ as follows:

$$\begin{aligned} Ax &= Ax' \\ Ay &= Ay' \cdot \sin\alpha - Az' \cdot \cos\alpha \\ Az &= Ay' \cdot \cos\alpha + Az' \cdot \sin\alpha \end{aligned} \quad \textbf{(2)}$$

The tilt angle can be easily estimated from a calibration step with the vehicle in standing position on a horizontal surface using the measured projection of gravity vector on smartphone z’ axis:

$$Az'\_cal = 1g \cdot \sin\alpha \quad \textbf{(3)}$$

being $g \cong 9.8\ m/s^2$ the earth gravity acceleration.

### *B. Fall detection*

Notice that using vehicle accelerations, the ideal conditions to be tested to detect the accidental vehicle fall are:

$$\begin{aligned} Az &\cong 0 \\ Ax^2 + Ay^2 &\cong (1g)^2 \end{aligned} \quad \textbf{(4)}$$

Note that Ax and Ay may have different values depending on the orientation of the handlebar where the smartphone is attached, for this reason the fall condition is expressed using the addition of both components squared. Appropriate tolerances have been added to account for non-ideal situations, such as accelerometer calibration errors or the fact that the fall may not result in a vehicle laying in a perfectly horizontal position.

### *C. Binary integration of fall decisions*

A preliminary requirement of accident detection probability has been set with Pd > 0.999 corresponding to missing less than one fall per year in large cities where the number of accidents reported is presently under 1000 per year. The probability of false accident detection has to be very small to avoid costly decisions such as mobilizing police or medical help. Given the number of scooter rides in a large city in

the order of 10000 rides/day, the probability of false alarm of an accident detector taking decisions every 0.1 s should be lower than $1.1 \cdot 10^{-9}$ to obtain an average time between false alarms reported to the city traffic authority of one day. These requirements have been satisfied by a safety margin of more than two orders of magnitude using a binary integration of single detections. Binary decision integration methods have been studied in radar systems resulting on improved target detection and false alarm reduction probabilities (Levanon, N., 1998), with robust performance in front of different noise statistics (Skolnik, M., 1990). In this way, an accident is declared if a riding vehicle provides at least 50 fall detections in a group of 100 successive acceleration measurements. With this approach, the experimental record of the detector is excellent, with no reported detection errors when the smartphone is properly installed in scooters. Similar performance is expected in bikes since the principle of operation is the same.

## *3. Conclusions and future work*

In this work we have presented two different algorithms, type of lane recognition and accident detection, which have been developed in the framework of the RideSafeUM project, which aims to bring micromobility safety benefits to users, public authorities and operators through the use of innovative technology. In the RidsSafeUM solution, safety is achieved by using two different mechanisms: information to the user and information to the authorities. The first mechanism is intended to avoid accidents by increasing the user's awareness of the local micromobility regulations. The second one aims at alleviating the consequences of accidents by detecting them automatically, and informing the authorities so that they can evaluate the severity of the accident and call the emergency services if necessary.

The first mechanism is implemented with the proposed computer vision classification algorithm. This algorithm can determine, accurately and in real-time, the type of lane using the user's smartphone. For the second mechanism, we have also presented the accident detection algorithm, which uses the accelerometer of the same smartphone. It can recognize whether an accident has occurred with high confidence. These two algorithms allow to increase the riding safety and help unleashing the full potential of micromobility.

***Acknowledgements***

We sincerely acknowledge the financial support that funding bodies have provided for this work. We thank the Spanish Ministry of Science, Innovation and Universities (PID2020- 117142GB-I00 [DeeLight project] by MCIN/AEI/10.13039/501100011033). Additionally, we extend our gratitude to the EIT Urban Mobility, an initiative of the European Institute of Innovation and Technology (EIT), a body of the European Union, for their support under the RideSafeUM project.

## *4. References-Bibliography*

Aloul, F., Zualkernan, I., Abu-Salma, R., Al-Ali, H., and Al-Merri, M.,(2014). *iBump: Smartphone application to detect car accidents*. In International Conference on Industrial Automation, Information and Communications Technology (IAICT), IEEE, 52–56.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). *Imagenet: A large-scale hierarchical image database*. In 2009 IEEE conference on computer vision and pattern recognition (pp. 248–255).

Goodfellow, I., Bengio, Y. and Courville, A. (2016). *Deep Learning*, MIT Press.

Hirawat, A., and Bhargava, D.,(2015). *Enhanced accident detection system using safety application for emergency in mobile environment: Safeme*. In Proceedings of Fourth International Conference on Soft Computing for Problem Solving, Springer (2015), 173–179.

Howard, A.G., *et al.* (2017). *Mobilenets: Efficient convolutional neural networks for mobile vision applications*. CoRR, abs/1704.04861.

Howard, A. *et al*., (2019). *Searching for MobileNetV3*. (2019) IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Korea (South), 2019, pp. 1314-1324

Iandola, F.N., Moskewicz, M.W., Ashraf, K., Han, S., Dally, W.J., and Keutzer, K (2016). *Squeezenet: Alexnet-level accuracy with 50x fewer parameters and < 1mb model size*. CoRR, abs/1602.07360

Kau, L.-J. and Chen, C.-S. (2015). *A Smart Phone-Based Pocket Fall Accident Detection, Positioning, and Rescue System*, in IEEE Journal of Biomedical and Health Informatics, vol. 19, no. 1, pp. 44-56

Lahn, J., Peter, H., and Braun, P., (2015). *Car crash detection on smartphones*. In Proceedings of the 2nd international Workshop on Sensor-based Activity Recognition and Interaction (iWOAR '15).

Levanon, N. (1988). *Radar Principles*, John Wiley & Sons, New York. Chapter: Radar Detection.

Ma, N., Zhang, X., Zheng, HT., Sun, J. (2018). *ShuffleNet V2: Practical Guidelines for Efficient CNN Architecture Design*. ECCV 2018.

Mateo, A. (2022). *Lane type classification using efficient networks*. Ms. Thesis. Universitat Politècnica de Catalunya (UPC).

Pedregosa *et al.*, (2011). *Scikit-learn: Machine Learning in Python*, JMLR 12, pp. 2825-2830. URL: https://scikit-learn.org/stable/modules/model_evaluation.html#multiclass-and-multilabel-classification

Puig, J. (2022). *Visual Determination of Lane Type for Micromobility*. Degree Thesis. Universitat Politècnica de Catalunya (UPC).

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A. and Chen, L.C. (2018). *MobileNetV2: Inverted Residuals and Linear Bottlenecks*. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 2018, pp. 4510-4520

Skolnik, M. (1990). *Radar Handbook*, 2nd Edition, McGraw Hill, New York. Chapter: Automatic Detection Tracking and Sensor Integration.

Snyder, J.S., Ferguson, D. and Irey, G, (2015). Automatic crash detection, U.S. Patent US9767625B1

Tan, M. and Le, Q.V. (2019) *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks*. Proceedings of the 36th International Conference on Machine Learning, ICML 2019, Long Beach, 9-15 June 2019, 6105-6114.

Watthanawisuth, N., Lomas, T., Tuantranont, A. (2012). *Wireless black box using mems accelerometer and GPS tracking for accidental monitoring of vehicles*. In: Proc IEEE international conference on biomedical and health informatics; p. 847–50.

White, J., Thompson, C., Turner, H., Dougherty, B., and Schmidt, D. C., (2011). *Wreckwatch, Automatic traffic accident detection and notification with smartphones*. Mobile Networks and Applications 16, 3, 285–303.

Zhang, X., Zhou, X., Lin, M. and Sun, J. (2018). *ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices.* 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018, pp. 6848-6856.