# Road Surface Type Classification Based on Inertial Sensors and Machine Learning: A Comparison Between Classical and Deep Machine Learning Approaches for Multi-Contextual Real-World...

**Article** *in* Computing · October 2021

DOI: 10.1007/s00607-021-00914-0

**2 authors:**

Jeferson Menegazzo
Federal University of Santa Catarina
**9** PUBLICATIONS **25** CITATIONS
SEE PROFILE

Aldo Von Wangenheim
Federal University of Santa Catarina
**539** PUBLICATIONS **2,148** CITATIONS
SEE PROFILE

Computing
https://doi.org/10.1007/s00607-021-00914-0

REGULAR PAPER

# Road surface type classification based on inertial sensors and machine learning

## A comparison between classical and deep machine learning approaches for multi-contextual real-world scenarios

Jeferson Menegazzo[1] · Aldo von Wangenheim[1]

Received: 1 August 2020 / Accepted: 28 January 2021

**Abstract**
The demand for several sources of situational data from the traffic environment has intensified in recent years, through the development of applications in intelligent transport systems (ITS), such as autonomous vehicles and advanced driver assistance systems. Among these situational data, the road surface type classification is one of the most important and can be used throughout the ITS domain. However, in order to have a wide application, the development of a safe and reliable model is necessary. Therefore, in addition to the application of safe technology, the model developed must operate correctly in different vehicles, with different driving styles and in different environments in which vehicles can travel to. For this purpose, in this work we collect nine datasets with contextual variations using inertial sensors, represented by accelerometers and gyroscopes. These data were produced in three different vehicles, with three different drivers, in three different environments in which there are three different surface types, in addition to variations in conservation state and presence of obstacles and anomalies, such as speed bumps and potholes. After a pre-processing step, these data were used in 34 different computational models for road surface type classification, employing both Classical Machine Learning and Deep Learning techniques. Through several experiments, we analyze the learning and generalization capacity of each technique. The best model developed was a CNN-based deep neural network, which obtained validation accuracy of 93.17%, classifying surfaces between segments of dirt, cobblestone or asphalt roads.

**Keywords** Road conditions · Road surface type · Intelligent transport systems · Inertial sensors · Pattern recognition · Machine learning

**Mathematics Subject Classification** 68Q07 · 68T05 · 68T07 · 68T10 · 68T40

Extended author information available on the last page of the article

Published online: 17 February 2021

## 1 Introduction

In Intelligent Transport Systems (ITS) sensors are used in the transport infrastructure [33] and in its participants [34]. Through these sensors, raw data are produced, on which Artificial Intelligence (AI) techniques are applied in order to generate situational information to assist decision-making [32]. This information has wide applicability: When it comes to the support of human decision-making, they can be applied in Advanced Traveler Information Systems (ATIS), Advanced Traffic Management Systems (ATMS), Advanced Public Transport Management Systems (APTMS), Advanced Driver Assistance Systems (ADAS) and others [46]. In order to support machine-based decision-making, this information can also be used by Intelligent Agents to coordinate the actions of autonomous vehicles.

Situational information produced in ITS can be classified in several ways. Through a state-of-the-art analysis in a comprehensive Systematic Literature Review (SLR) [35] and systematic mapping of approaches and methods [36] , we established the concept of *vehicle perception*: Sensors are installed in the vehicle in order to produce exteroceptions and proprioceptions. Exteroceptions aim at understanding the environment outside the vehicle, allowing it to recognize path features. These features include transient events in the form of anomalies and obstacles, such as potholes, cracks, speed bumps, etc.; and persistent events, such as surface type, conservation condition and the road surface quality. On the other hand, proprioception focuses on understanding vehicular events in order to allow a vehicle to identify their own behavior. These can also be transient in the form of driving events, such as lane change, braking, skidding, aquaplaning, turning right or left; and persistent, such as a safe or dangerous driving behavior profile [35,36].

Two main sensing approaches were found in the literature for the collection of raw data that generate information in the form of vehicular perception after processing by AI techniques: Active and passive sensing [35,36]. Active techniques require interaction with the environment in order to produce raw data, such as emitting signals into the environment through LIDAR or radar. Passive techniques do not require active interaction with the environment, passively sampling physical or visual data. Therefore, they are considered safer, non-polluting and generally of lower cost. These features turn passive techniques into more interesting solutions for large-scale use, such as autonomous vehicles [35,36]. Among the existing passive technologies are inertial sensors, which are based on the principle of inertia for data production [8]. Represented by gyroscopes and accelerometers, these devices produce, respectively, one-dimensional signals of rotation rate and acceleration force in three physical axes [18]. In the present study, these signals are the result of vehicle traction and interactions with the environment in which it travels.

The application of inertial sensors for the production of vehicular perception has been studied in recent years, in all types of transient and persistent events mentioned above [35,36]. However, in relation to the nature of the environment in which such perception occurs, the vast majority of previous studies focus only on the recognition of potholes or speed bumps, and on the quantification of road surface quality indexes. Surface type classification has been little explored, constituting an important perception type to be improved, especially if we consider that cars equipped with ITS are becoming more widespread and finding application in situations where unpaved roads and badly maintained pavements are common and intelligent vehicles will have to be able to sense and adjust their driving modes to such environments. The availability of this situational information can benefit a large number of applications. Production flow and logistics systems can use this information as a criterion for planning routes that do not increase the production cost, since the surface type impacts on fuel consumption, travel time and possible damage to vehicles. ADAS-type systems can use it to warn the driver of the pavement change on the way, as well as allowing to consult the route before the travel. Government systems can also use this information as criteria to define priority in road maintenance, and control the traffic flow based on the pavement type or the lack of it. In autonomous vehicles, this data can be used in order to validate information produced by other sensors, making the intelligent system more robust for decision-making. For example, the road surface type can be used as an evidence to validate hypotheses raised by computer vision.

Although the road surface type classification produced from inertial sensors can be used in the most varied applications, few studies currently use this data type. We understand that this is partly due to the lack of a reliable and adaptable model. This fact is evidenced by the SLR [35,36], where we observed that the development of an adaptive model constitutes the main research gap, since there are several dependency factors that interfere in the analyzed data. These dependency factors are related to the existing contextual diversity, where the model can be applied to different vehicles, drivers and environments. In order to allow a wide adoption of a road surface type classification model, we understand that adaptability is an essential feature, where the developed solution needs to present a certain degree of reliability when applied in uncontrolled contexts, such as real-world scenarios.

With this objective in mind, in this work we developed and tested 34 models for road surface classification considering three stages. In the first stage, *data collection*, we collect several datasets with contextual variations, including unpaved roads, asphalt and cobblestone-paved roads, collected on different cars and by different drivers. In the second stage, *pre-processing*, we adjust and separate the data in experiments that have the objective to assess the generalization ability of each technique for unknown context. In the third stage, *processing*, we apply different Classical and Deep Learning techniques to assess the most appropriate approach.

This paper is organized in 8 sections as follows: in Sect. 2 we discuss the dependency factors to be considered in the modeling. In Sect. 3 we present the related works, detailing what has already been done and what differs in this study from the previous ones. In Sect. 4 we detail the methodology for data sampling, from the sensors positioning, their placement in the vehicle structure and reference frames for collection and analysis. In Sect. 5, the data pre-processing is presented in order to adjust the signals before being used in the classification technique. In Sect. 6 we detail the Machine Learning methods experimented, among classical machine learning and deep learning techniques. In Sect. 7, the results obtained are discussed and compared. Finally, in Sect. 8 we present the final considerations and our suggestions for future work.

## 2 Dependency factors

Through our literature review analysis [35,36] and the conducting exploratory experiments, we observed the existence of dependency factors that affect the application of the solution in real-world scenarios. These factors should be considered in the development of a road surface type classification model through inertial sensors, so the proposed solution can recognize the patterns in different conditions. These factors were classified in the form of four dependency properties described below.

*Sensory Properties:* consist of the sensor orientation [27], measurement range and sampling rate. Since the inertial sensors have their own reference frame and the data analysis must be done in the vehicle's reference frame, the sensor orientation must be considered. Therefore, the sensor must be placed aligned with the vehicle's coordinate system or have an axes reorientation to that reference frame. Regarding the measurement range, it is necessary that the sensor has an adequate range in order to be able to sample the data without saturation, since in this study scenario large measurements are produced. Finally, the sampling rate must take into account not only the computational cost, but also if there will be enough samples to perform the classifications at a given speed.

*Vehicular Properties:* are related to vehicular structure, where the main factor is the vehicle suspension system [27,53]. In order to smooth the impacts caused by the road surface, this system causes the values to be reduced when measured above the suspension, as in the dashboard. Thus, the values sampled below the suspension have interference of the unsprung mass through the tire stiffness and tire absorptivity, and those sampled above the suspension add the influence of the sprung mass, through spring and shock absorber [2]. Thus, to carry out the classifications in different vehicles, we must consider that each one has different characteristics related to the sprung and unsprung mass, and the sensor location in the vehicle structure may have more or less interference from these properties.

*Driving Properties:* are related to the vehicle driver, these properties refer to the different driving styles. Each driving style produces different impacts on the sampled signals, according to the applied traction and direction and the way the vehicle interacts with the environment. The main factor to be considered is the longitudinal speed [10,14,19,27,30,31,37,47], which impacts on the amplitude of the sampled signals and their distribution over time. Therefore, depending on the applied speed, the signals from the same road segment may have a greater or lesser amplitude and more or less samples, but it must be recognized in the same way.

*Environmental Properties:* relate to the characteristics of the study environment. Thus, in the proposed classification, a certain surface type has variations in the pavement conservation state, in the presence of potholes, speed bumps, other anomalies and obstacles. These variations should not influence the classification. For example, a poorly maintained asphalt with several anomalies must be classified equally to one in good condition.

Based on these properties, in order to develop and validate the reliability of a model for the road surface type classification, we observed the need for the research datasets to present variations of all dependency factors. Therefore, we must sample data in the most diverse contexts and consider these conditions in the subsequent processing steps for pattern recognition.

## 3 Related works

Based upon our SLR [35,36] and on the dependency properties that we identified in this SLR, in this section we analyze the models proposed in related research in order to verify which properties were adopted. The studies about road surface type classification using inertial sensors we were able to identify were conducted in two vehicle types: wheeled ground robots [23,43,49] and cars [48,51,52]. Since the structure between the two vehicle types is very different, and there is a vehicle dependency property, the papers using wheeled ground robots were not considered. Among the studies that apply sensors in cars, Ref. [48] used sensors embedded in smartphones and [51,52] attached them to the vehicle.

In [48] the smartphone with sensors was fixed inside the vehicle using a flexible suction holder near the dashboard. The model uses data from the accelerometer (100 Hz) and speed from Global Positioning System (GPS). The study obtained the best results with the model of Longest Common Subsequence Similarity (LCSS) combined with the Complexity Invariant Distance (CID). The road surface was classified as asphalt/flexible pavement (98.28%), cobblestone streets (84.41%) and dirt roads (78.64%), obtaining an average accuracy of 87.77%. In [51] and [52], data from the accelerometer mounted on the vehicle suspension and GPS speed were used. In pre-processing, the mathematical model Quarter Car (QC) was applied, in order to consider the influence of vehicle suspension, and Fast Fourrier Transform (FFT) for the feature extraction in frequency domain. These data were trained on a Support Vector Machine (SVM), classifying the road in asphalt (17.6%), concrete (99.6%), grass (74.9%) and gravel (85.3%), with an average accuracy of 69.4%.

Through the related studies we observed that the results obtained demonstrate low values of evaluation metrics even in controlled contexts. Dependency factors are poorly explored, and there is no intention to analyze the models in contextual variations, limiting themselves to the use of accelerometer data, speed and the QC model. Therefore, we developed a methodology in this study that includes the dependency factors in three stages, which are data collection, pre-processing and processing. In data collection, we aim to build datasets with variations in dependency factors, so that we can validate the hypothesis of this study. Therefore, this study has the hypothesis that through the sampled and pre-processed data, Machine Learning techniques are able to learn the relationship and the influence of dependence factors on the sensor signals, in order to classify correctly the road segments and with greater values of evaluation metrics than it was presented in previous studies.

## 4 Data collection

Through the SLR [35,36] we did not find any publicly available datasets that provide data from inertial sensors and auxiliary sensors in this ITS context. Therefore,

**Table 1** Sensor network hardware

| Hardware | Sensor | Data | Sampling rate (Hz) |
|---|---|---|---|
| HP Webcam HD-4110 [41] | Camera | 720p Video | 30 |
| Xiaomi Mi 8 | GPS | Speed in m/s, latitude, longitude, etc. | 1 |
| MPU-9250 | Accelerometer | 3-Axis acceleration in $\mathrm{m/s^2}$ | 100 |
| | Gyroscope | 3-Axis rotation rate in deg/s | |
| | Magnetometer | 3-Axis ambient geomagnetic field in $\mu$T | |
| | Temperature | Sensor temperature in °C | |

several data collections were performed. Nine datasets named Passive Vehicular Sensors Dataset (PVS 1-9) were created, gathering data from several passive approach sensors. For data collection, two sensor networks were used, each one consisting of a Single-Board Computer (SBC) Raspberry Pi and three MPU-9250 modules, each one equipped with an accelerometer, a gyroscope, a magnetometer and a temperature sensor. An external GPS source was also used, producing location and speed data, as well as a camera for capturing ambient video. Table 1 details the hardware used.

All the hardware equipment was attached to the vehicle as shown in Fig. 1. The camera was placed on the outside car roof (1), and the GPS receiver was placed internally on the dashboard (2). The two networks with MPU-9250 modules were distributed in the vehicle in order to consider the data coming from points with different influences of the vehicular dependency property. Thus, each end of the front axle (right and left side) received one of the sensor networks, where a module was attached to the control arm (4), located below and close to the vehicle suspension; another module was placed above and close to the suspension, attached to the bodywork immediately above the tire (3); and another module was attached to the vehicle dashboard (2), inside the cabin. In relation to the MPU-9250 module sampling reference frame, the controlled positioning approach was used, where the module placement was performed in such a way that the three axes of the sensor coordinate system are aligned with the vehicle ones, being both a reference frame for collection and analysis. The accelerometer was set with a full scale of 8g[1] and the gyroscope with 1000 deg/s, being consistent ranges to not saturate (sensory property).

[1] $1\mathrm{g} = 9.8\ \mathrm{m/s^2}$.

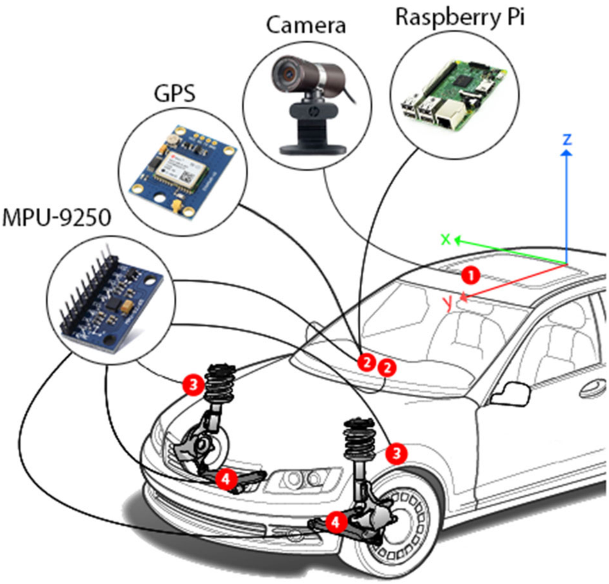


**Fig. 1** Structure of the sensor network in the vehicle. *Source* the authors

**Table 2** Datasets description

| Dataset | Vehicle | Driver | Scenario |
|---|---|---|---|
| PVS 1 | Volkswagen Saveiro | Driver 1 | Scenario 1 |
| PVS 2 | Volkswagen Saveiro | Driver 1 | Scenario 2 |
| PVS 3 | Volkswagen Saveiro | Driver 1 | Scenario 3 |
| PVS 4 | Fiat Bravo | Driver 2 | Scenario 1 |
| PVS 5 | Fiat Bravo | Driver 2 | Scenario 2 |
| PVS 6 | Fiat Bravo | Driver 2 | Scenario 3 |
| PVS 7 | Fiat Palio | Driver 3 | Scenario 1 |
| PVS 8 | Fiat Palio | Driver 3 | Scenario 2 |
| PVS 9 | Fiat Palio | Driver 3 | Scenario 3 |

This work aims to develop a model that considers the dependency factors, in addition to the speed sampling (driving properties) and the placement of inertial sensors in different points of the vehicle body (vehicular property), it is necessary to produce data in different contexts, to obtain a variability of scenarios to validate the model. Therefore, the set of sensors detailed above was used in three different vehicles (vehicular property), with three different drivers varying speed from 0 to 91.98 km/h (driving property) covering three different scenarios (environmental property), where each one has three different surface types. Each scenario also has general environment variations, such as the presence of speed bumps, potholes, variations on pavement conservation state, etc. Table 2 details the context of each dataset.

Regarding the road surface type, each scenario contains unpaved (dirt roads) and paved road segments (asphalt or cobblestone roads). Figure 2 illustrates each scenario,

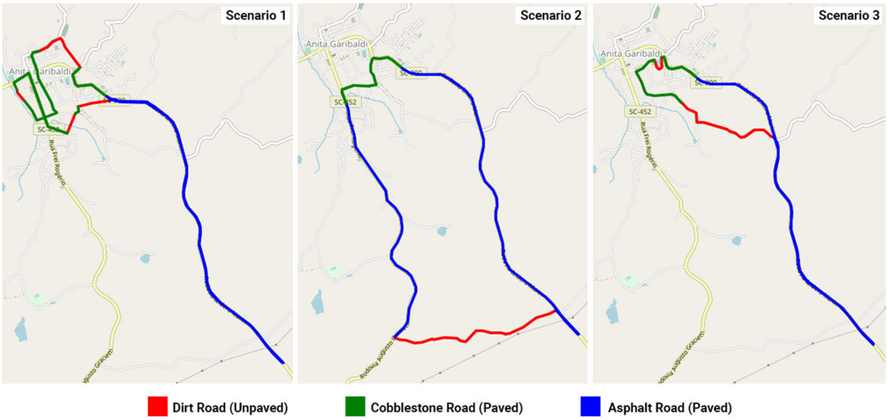
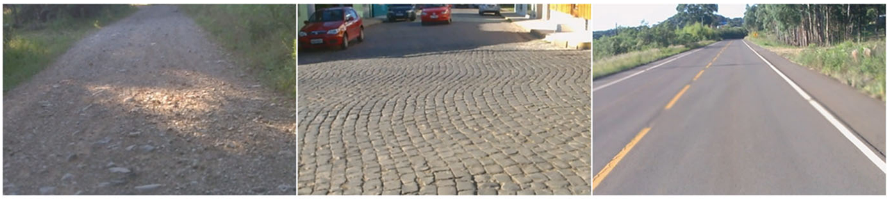


**Fig. 2** Data collection scenarios. *Source* the authors



Dirt Road (Unpaved) Cobblestone Road (Paved) Asphalt Road (Paved)

**Fig. 3** Surface types in scenarios. *Source* the authors

Fig. 3 illustrates the surfaces and Table 3 quantifies samples and distances. The data were collected in the municipality of Anita Garibaldi, in upstate Santa Catarina, Brazil, between December 24th and 26th, 2019. The midpoint of the scenarios is given by the coordinates (– 27.69983094872972, – 51.11577673365139). The source-codes used in the data collection for the sensors manipulation, sampling and storage of the signals; in pre-processing, with adjustments, data combination, normalization, etc.; in processing, with the pattern classification; as well as the datasets themselves, are documented and available on Github for future research.[2]

## 5 Pre-processing

After the collection, the raw data were pre-processed to be used in the machine learning techniques, according to the steps illustrated in Fig. 4. Initially, we made data adjustments, such as combining data from inertial sensors with those from GPS, where for each sample of the MPU-9250, the last known location and speed was associated. Videos of plotting the signals were also created, available on Github, which were

[2] https://github.com/Intelligent-Vehicle-Perception/Intelligent-Vehicle-Perception-Based-on-Inertial-Sensing-and-Artificial-Intelligence and https://codigos.ufsc.br/lapix/intelligent-vehicle-perception-based-on-inertial-sensing.

**Table 3** Datasets metrics

| Scenario | Dataset | Samples (no.) | | | | Distance (km) | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Dirt road | Cobblestone road | Asphalt road | Total | Dirt road | Cobblestone road | Asphalt road | Total |
| 1 | PVS 1 | 25,868 | 61,659 | 56,509 | 144036 | 1.59 | 3.53 | 8.7 | 13.81 |
| | PVS 4 | 23,903 | 57,670 | 50,919 | 132,492 | 1.58 | 3.51 | 8.72 | 13.81 |
| | PVS 7 | 23,778 | 54,224 | 50,546 | 128,548 | 1.59 | 3.49 | 8.69 | 13.78 |
| 2 | PVS 2 | 44,618 | 20,737 | 59,330 | 124,684 | 2.17 | 1.39 | 8.07 | 11.62 |
| | PVS 5 | 60,539 | 18,143 | 55,195 | 133,877 | 2.16 | 1.38 | 8.09 | 11.63 |
| | PVS 8 | 44,939 | 18,825 | 59,854 | 123,618 | 2.16 | 1.37 | 8.09 | 11.63 |
| 3 | PVS 3 | 28,659 | 26,143 | 51,014 | 105,816 | 1.66 | 1.66 | 7.4 | 10.72 |
| | PVS 6 | 23,888 | 31,641 | 40,750 | 96,279 | 1.38 | 1.93 | 7.43 | 10.73 |
| | PVS 9 | 23,153 | 25,182 | 43,220 | 91,555 | 1.67 | 1.66 | 7.42 | 10.74 |

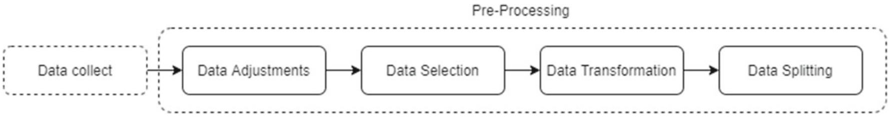


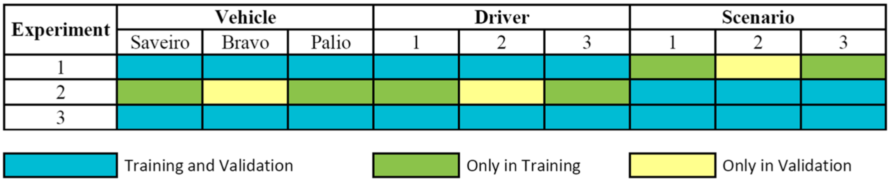
**Fig. 4** Pre-processing steps. *Source* the authors

synchronized with the ambient video and it helped in the creation of the surface type labels in the one-hot-encoded form. Therefore, this study has a human Ground-Truth (GT).

The data to be used were selected after the initial adjustments. This study focused on the use of the values sampled in the control arm, located below and close to the suspension system. According to the Quarter Car (QC) model, this values have interference on the unsprung mass through the tire stiffness and tire absorptivity [2]. Therefore, each sample used has the acceleration force and rotation rate values, both on three axes, and the speed value. In this study, we considered all axes and not just the one of greatest interest, as is usually done in studies of exteroception with inertial sensors [35,36], since all axes present important information, which should be considered in the development of a model with greater accuracy, precision and recall values.

After selection, the data were transformed to suit the inputs of the pattern classification techniques. For Classical Machine Learning methods, data features were extracted using the statistical methods Standard Deviation, Mean and Variance for inertial sensors signals, and Mean for speed, which are the methods most commonly used in related studies to extract high-level vibration-based features on data from inertial sensor signals [4–7,21,29,30,39,40,42,47]. To analyze the influence of the number of samples, fixed windows of 100, 150, 200, 250 and 300 samples were used, considering 100 the minimum size to have enough information, and 300 the maximum size so that the delay of the first classification is not greater than 3 s. For Deep Learning methods, the data were normalized with Robust Scaler, Min Max Scaler (0, 1) and Min Max Scaler (− 1, 1), since these techniques produce better performance when the variables are scaled over a values range. Afterwards, the data were also grouped in windows of the same size as the classical techniques.

After the transformation, we define what data would be used for training and validation. Although a same dataset is commonly divided into part for training and another for validation (70–30%), or the datasets are applied in a k-fold cross-validation methodology, both approaches in the context of this study incurs in a bias for two reasons. The first, if all datasets have a part for training and another for validation or are distributed in the k-fold approach of k equal to 9, this implies that the pattern recognition technique will have training data values sampled on all vehicles, with all drivers in all scenarios. This, in itself, already leads the technique to obtain excellent results, but it does not tell us anything about the ability to generalize the pattern classification in unknown contexts. Second, this type of splitting can lead to situations in which the validation data is mostly made up of data with more easily recognized patterns, such as asphalt pavement segments, since the signal vibration in this surface type is much less than in the others. Therefore, in order to correctly assess the generalizability of each technique, assessing their reliability in unknown contexts (vehicle, driver or sce-



| Experiment | Vehicle | | | Driver | | | Scenario | | |
|---|---|---|---|---|---|---|---|---|---|
| | Saveiro | Bravo | Palio | 1 | 2 | 3 | 1 | 2 | 3 |
| 1 | 🟦 | 🟦 | 🟦 | 🟦 | 🟦 | 🟦 | 🟩 | 🟨 | 🟩 |
| 2 | 🟩 | 🟨 | 🟩 | 🟩 | 🟨 | 🟩 | 🟦 | 🟦 | 🟦 |
| 3 | 🟦 | 🟦 | 🟦 | 🟦 | 🟦 | 🟦 | 🟦 | 🟦 | 🟦 |

🟦 Training and Validation 🟩 Only in Training 🟨 Only in Validation

**Fig. 5** Dataset division into training and validation groups. *Source* the authors

nario), we decided to divide the training and validation data according to the dataset, separating them into three specific experiments:
*Experiment 1:* The model learns data from all vehicles and drivers for some scenarios; but not all vehicles with all drivers for all scenarios.

- Train (65%): PVS 1, PVS 3, PVS 4, PVS 6, PVS 7, PVS 9.
- Validation (35%): PVS 2, PVS 5, PVS 8.

*Experiment 2:* The model learns data from all scenarios for some vehicles and some drivers; but not all vehicles with all drivers for all scenarios.

- Train (66%): PVS 1, PVS 2, PVS 3, PVS 7, PVS 8, PVS 9.
- Validation (34%): PVS 4, PVS 5, PVS 6.

*Experiment 3:* The model learns data from some vehicles with some drivers for some scenarios, but not all vehicles with all drivers for all scenarios.

- Train (66%): PVS 1, PVS 2, PVS 4, PVS 6, PVS 8, PVS 9.
- Validation (34%): PVS 3, PVS 5, PVS 7.

It is important to note that each dataset consists of a disjoint set of variables. Therefore, each dataset contains data for all model input variables (acceleration force, rotation rate, and speed) and with the representation of all data classes (dirt, cobblestone, and asphalt). What differentiates one dataset from another is the data collection context, where there is variation in the dependency properties. Figure 5 illustrates the division of training and validation groups according to these properties.

Although the data division approach adopted is not common, it was necessary so that we could evaluate the behavior of the Machine Learning models when applied to data collected in an unknown vehicle, driver, or scenario. In the three experiments, all data classes are present in the training and validation phase. To assess the need to apply data class balancing techniques, we measure the distribution of each class as detailed in Table 4. We calculate the class distribution for the training data [20,26], since these data are used for models to learn. The distribution was calculated in relation to the number of samples, since the data windows are defined according to this.

Analyzing the Table 4, we have that the class distribution is between 21.36 and 44.61%, depending on the experiment. The ratio varies between 1:1.2 and 1:3.7, where for 1 sample of a certain data class, there are from 1.2 to 3.7 samples in the other classes. According with [15], a dataset is said to be imbalanced when there is a significant, or in some cases severe, disproportion among the number of samples of each class. Class imbalance can be considered slight or severe, where imbalance ratios ranging from 1:4 up to 1:100 (20–1% distribution) are considered slight imbalance and imbalance

**Table 4** Distribution of data classes

| | Data class | | | | | |
|---|---|---|---|---|---|---|
| | Dirt road | | Cobblestone road | | Asphalt road | |
| | Percentage | Ratio | Percentage | Ratio | Percentage | Ratio |
| All datasets | 27.69 | 1:2.6 | 29.07 | 1:2.4 | 43.23 | 1:1.3 |
| Experiment 1 Train | 21.36 | 1:3.7 | 36.71 | 1:1.7 | 41.92 | 1:1.4 |
| Experiment 2 Train | 26.59 | 1:2.8 | 28.78 | 1:2.5 | 44.61 | 1:1.2 |
| Experiment 3 Train | 26.15 | 1:2.8 | 30.26 | 1:2.3 | 43.58 | 1:1.3 |

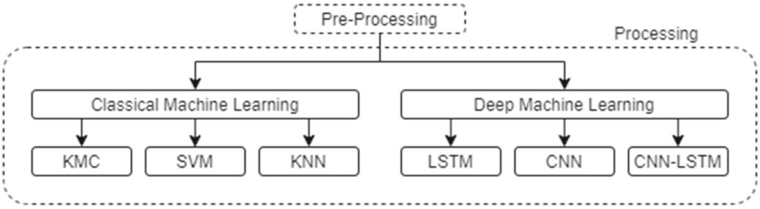


**Fig. 6** Approaches and techniques for surface type classification. *Source* the authors

ratios ranging from 1:100 or more ($<$ 1% distribution) are considered severe imbalance [9,25]. As we can observe, the distribution ratio of data classes in this study is not even classified as slight imbalance, since in the worst case, the ratio 1:3.7 is still a more uniform distribution than 1:4. Since the slight imbalance is often not a concern, and the problem can often be treated like a normal classification predictive modeling problem [9], there is no need to apply class balancing in this study.

## 6 Processing

After the pre-processing step, the data were used in Artificial Intelligence techniques to classify road surface type. Through our SLR on vehicle perception [35,36], we have identified the main techniques used in vehicle exteroception with inertial sensors, where the objective is to recognize the characteristics of the external environment such as a surface type, pavement quality, potholes, speed bumps, etc. All machine learning models used in previous studies in the field are based on classical techniques. Therefore, in this study we propose a comparison between the most commonly used techniques in the area, all of these being Classical Machine Learning, with Deep Learning techniques, which have not yet been used in these problems. Thus, we use the approaches and techniques illustrated in Fig. 6 to identify the most appropriate. In the following sections, we detail the developed models.

## 6.1 Classical machine learning

Several Classical Machine Learning techniques were used in related studies about exteroception with inertial sensor signals [35,36]. As previously discussed, in order to apply these methods, instead of the Deep Learning approach, classical techniques require pre-processing step to extract high-level features from the raw data, which well represent the patterns to be recognized [17]. Among the methods with this approach most applied in the area, there are the three experimented in this study: K-Means Clustering (KMC), Support Vector Machines (SVM) and K-Nearest Neighbors (KNN).

### 6.1.1 K-means clustering—KMC

The KMC consists in an unsupervised technique for clustering, which allows the identification of similar data clusters. In this method, each data is assigned to one of the k clusters based on a metric distance in relation to the cluster's centroid. The centroid of each clusters is iteratively recalculated based on the data average, and the distances recalculated, until the centroids are stabilized, or the maximum number of iterations are reached [16,38]. For this study, the KMC models were defined with the number of 3 clusters, representing the classes of road surface type to be classified.

### 6.1.2 K-nearest neighbors—KNN

The KNN consists in a classification technique based on similarity metrics between data to recognize patterns. In this way, for a new data, the distance between the data and each one of the training data is calculated, identifying the k closest neighbors. The new data class is defined as the most common class among its k neighbors [24]. In order to identify the optimal neighbors value for this study, models with 1, 2, 5, 10, 50, 100, 250, 500, 1000 neighbors were tested.

### 6.1.3 Support vector machines—SVM

The SVM is a supervised learning technique applied in classification, where the algorithm searches for an optimal hyperplane that separates the data classes. In non-linear problems it is necessary to use the kernel hyperparameter, which converts a non-separable problem into a separable problem, so that SVM can classify it [45]. In this study, models for 3 kernels were analyzed: polynomial with degree 3, rbf and sigmoid.

## 6.2 Deep learning

In the application of Classical Machine Learning techniques, the features extraction that well represent the data to be classified is a central problem. Therefore, in many application domains, it is difficult to extract high-level features through raw data [17]. With the Deep Learning development, this problem was solved, making it possible to build computational models composed of multiple processing layers to learn the data representation with different abstraction levels [28]. Therefore, layered representations were introduced, which define representations that are expressed in terms of other, building complex concepts through simple concepts [17].

With this new approach, the state of the art in speech recognition, visual object recognition, object detection and many other applications have been drastically improved [28]. In studies of vehicular perception with inertial sensors, few use this approach to recognize patterns in vibration signals, but none of the found studies apply these techniques to the road surface classification [35,36]. Thus, the Deep Neural Networks (DNN) developed were based on studies in human activity recognition [1,3,11,12,50,54,56,57], walking speed estimation [44] and outdoor terrain types classification during running [13]. We build and analyze LSTM-based, CNN-based and CNN-LSTM-based models. All models developed use the Adam optimizer in conjunction with the loss function Categorical Cross Entropy, since this is a multiclass classification problem.

### 6.2.1 Long short-term memory—LSTM

LSTM is a type of Recurrent Neural Network (RNN) considered ideal for prediction and classification of time series [55]. In this technique, the concept of memory cell and auto-loops was introduced, where the network can maintain values for a short or long period of time, as a function of its inputs. Thus, the cell can remember what is important, and not just the last computed value [22], so the relevant information from previous entries is retained and used to change the current output [57]. For the LSTM-based approach, several network models were developed, among Vanilla LSTM and Stacked LSTM, both in unidirectional and bidirectional forms, detailed in Table 5.

LSTM 7 was the model developed with the best results, constituting a Stacked Unidirectional LSTM detailed in Fig. 7. In this model, DNN receives an input tensor $windows \times sequences \times features$, where *windows* are the groupings of windows, *sequences* are the data that a window has, and *features* are the values of the 7 sensor variables. The model is composed of an input layer, three recurrent and regularization blocks, and a block of fully connected layers for output production. LSTM layers of 100 units are used to learn the long-term temporal dependencies between time steps of data sequence. Regularization is done by two layers, Batch Normalization to standardize the inputs of a new layer, reducing training time [57] and improving performance; and Dropout by 50% to avoid overfitting by ignoring randomly selected neurons during training. After processing in the recurring layers, the parameters go to two Dense layers, the first with 100 neurons with Relu activation, and the second with 3 neurons and Softmax activation, producing the expected classification. As the expected output from the network, we use the labels corresponding to the last sample in the window.

### 6.2.2 Convolutional neural network—CNN

CNN is a type of neural network capable of extracting features directly from signals, through convolutions [13]. When applied to the classification of time series, CNN has two advantages over other techniques: local dependence, since nearby signals are probably correlated; and scale invariance for different paces or frequencies [50]. In this

**Table 5** LSTM-based DNN models

| Name | layers |
|---|---|
| LSTM 1 | 1 LSTM with 100 units, 1 Dropout of 0.5, 1 Dense of 100 units and relu activation<br>1 Dense with 3 units and softmax activation |
| LSTM 2 | Same as LSTM 1, but connecting LSTM returned sequences flattened directly to Dense block |
| LSTM 3 | Same as LSTM 1, but LSTM are bidirectional |
| LSTM 4 | Same as LSTM 2, but LSTM are bidirectional |
| LSTM 5 | 3 blocks of LSTM with 100 units and Dropout 0.5, 1 Dense with 100 units and relu activation<br>1 Dense with 3 units and softmax activation |
| LSTM 6 | Same as LSTM 5, but LSTM are bidirectional and Dropout 0.2 |
| LSTM 7 | 3 blocks of LSTM with 100 units, Batch Normalization and Dropout 0.5<br>1 Dense with 100 units and relu activation, 1 Dense with 3 units and softmax activation |

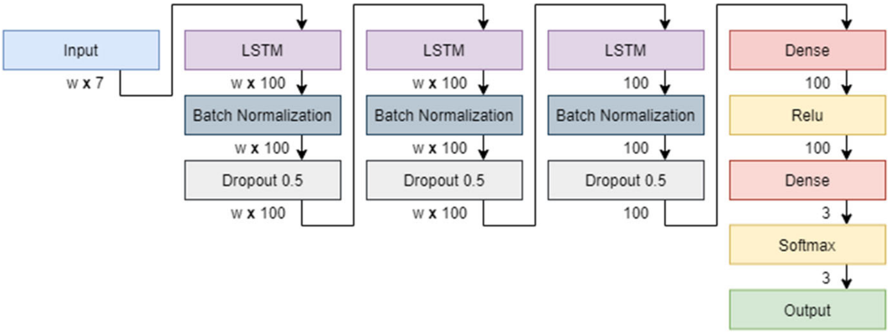


**Fig. 7** Best LSTM-based DNN model. *Source* the authors

type of DNN, each convolution layer has a $n$ number of filters applied with a kernel of size $m$. After convolution, there are usually pooling and fully connected layers, which perform classification tasks [50]. For the CNN-based approach, CNN Stacked models were produced using different pooling layers, it is detailed in Table 6.

The CNN-based model developed with the best results was CNN 8, detailed in Fig. 8. The DNN receives an input tensor $windows \times sequences \times features$ similar to LSTM-based, which are processed by three blocks of convolution and regularization layers,

**Table 6** CNN-based DNN models

| Name | Layers |
|---|---|
| CNN 1 | 3 blocks of Conv1D with 64-64-128 filters, kernel 3 and relu activation, 1 Flatten, 1 Dense with 100 units and relu activation, 1 Dense with 3 units and softmax activation |
| CNN 2 | Same as CNN 1, but with a max pooling 1D of pool size 2 after the convolution blocks |
| CNN 3 | 1 Conv1D with 64 filters, kernel 3 and relu activation, 1 Batch Normalization, 1 Dropout 0.15, 1 Conv1D with 32 filters, kernel 3 and relu activation, 1 Global Avg Pool 1D, 1 Batch Normalization, 1 Dropout 0.2 and 1 Dense with 3 units and softmax activation |
| CNN 4 | 2 Conv1D with 100 filters, kernel 10 and relu activation, 1 Max Pooling 1D with pool size 3 2 Conv1D with 160 filters, kernel 10 and relu activation, 1 Global Avg Pool 1D, 1 Dropout 0.5 and 1 Dense with 3 units and softmax activation |
| CNN 5 | 1 Conv1D with 64 filters, kernel 3 and activation relu, 1 Max Pooling 1D with pool size 2, 1 Conv1D with 64 filters, kernel 3 and activation relu, 1 Flatten and 1 Dense with 3 units and softmax activation |
| CNN 6 | 1 Conv1D with 24 filters, kernel 8 and activation relu, 1 Batch Normalization, 1 Spatial Dropout 0.15, 1 Conv1D with 12 filters, kernel 8 and activation relu, 1 Global Avg Pool 1D, 1 Batch Normalization, 1 Dropout 0.2, 1 Dense with 48 units and relu activation, 1 Batch Normalization, Dropout 0.25, 1 Dense with 3 units and softmax activation |
| CNN 7 | 3 blocks of Conv1D with 128-64-32 filters, kernel 3, relu activation, Batch Normalization and Max Pooling 1D with pool size 2, 1 Flatten, 1 Dropout 0.2, 1 Dense with 3 units and softmax activation |
| CNN 8 | 2 blocks of Conv1D with 64-32 filters, kernel 3, relu activation, Batch Normalization and Dropout 0.15, 1 Conv1D 32 filters, kernel 3, relu activation, 1 Global Avg Pool 1D, 1 Batch Normalization, 1 Dropout 0.2, 1 Dense with 32 units and relu activation and 1 Dense with 3 units and softmax activation |

and a block of fully connected layers. The features extraction blocks use convolution kernels with size 3 on signals, the first with 64 filters, and the others with 32 filters. Regularization is done by the Batch Normalization and Dropout layers at 15% and 20%. The last block of features extraction also has a Global Average Pooling 1D layer to extract more robust features by mean values in each region, accelerating the training process and avoiding overfitting [50,54]. Finally, the last block consists of two Dense layers, one with 32 neurons and Relu activation, and the other 3 neurons and Softmax

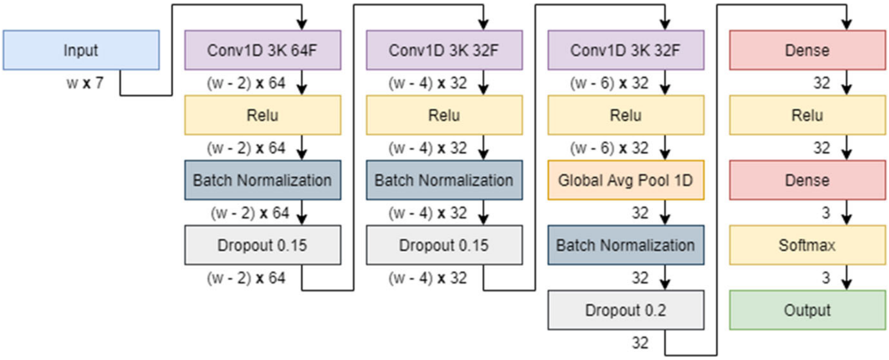


**Fig. 8** Best CNN-based DNN model. *Source* the authors

activation. As an expected output from the network, we use the most frequent labels present in the analyzed window.

### 6.2.3 CNN-LSTM

CNN-LSTM-based networks are hybrid models that use both convolution and recurrent layers [12]. In these DNNs, the signals are initially grouped into data windows, and each window in smaller blocks. Therefore, each block is processed by convolution layers that extract complex features [12]. Then, these blocks are delivered to the recurrent layers, to interpret the temporal sequence of these features. For this approach, we developed the models detailed in Table 7.

The CNN-LSTM-based model developed with the best results was the CNN-LSTM 6, detailed in Fig. 9. The DNN receives a tensor $windows \times sequences \times subsequences \times features$, where the *windows* are the groupings of windows, *sequences* are the data windows, *subsequences* are the window subgroups, and *features* are the values of the 7 sensor variables. The data is initially processed by three blocks of convolution and regularization layers, three blocks of recurrent and regularization layers, and a block of fully connected layers. The feature extraction blocks use convolution kernels with size 3 and 128 filters in the signals. Regularization is done by the Batch Normalization and Spatial Dropout 1D layers at 15% and 20%. The last block has a Global Average Pooling 1D layer. Then, the recurrent blocks have layers of LSTM with 100 units, and regularization by Batch Normalization and Dropout 50%. Finally, the resulting parameters are delivered to a Dense layer with 3 neurons and Softmax activation, producing the classification. As an expected output from the network, we use the labels corresponding to the last sample in the window.

**Table 7** CNN-LSTM-based DNN models

| Name | Layers |
|---|---|
| CNN-LSTM 1 | 2 Conv1D with 64 filters, kernel 3 and relu activation, 1 Dropout 0.5, 1 Max Pooling 1D with pool size 2, 1 Flatten, 1 LSTM with 100 units, 1 Dropout 0.2, 1 Dense with 100 units and relu activation, 1 Dense with 3 units and softmax activation |
| CNN-LSTM 2 | Same as CNN-LSTM 1, but with 3 Conv1D layers |
| CNN-LSTM 3 | Same as CNN-LSTM 2, but with 3 blocks of LSTM and Dropout after the Max Pooling 1D |
| CNN-LSTM 4 | Same as CNN-LSTM 3, but LSTM blocks do not have Dropout |
| CNN-LSTM 5 | 2 blocks of Conv1D with 128 filters, kernel 3 and relu activation, Max Pooling 1D with pool size 2 and Batch Normalization, 1 block of Conv1D with 128 filters, kernel 3 and relu activation, Global Avg Pool 1D and Batch Normalization, 3 blocks of LSTM with 100 units, Batch Normalization and Dropout 0.3, 1 Dense with 3 units and softmax activation |
| CNN-LSTM 6 | 2 blocks of Conv1D with 128 filters, kernel 3 and relu activation, Batch Normalization, and Spatial Dropout 0.15, 1 block of Conv1D with 128 filters, kernel 3 and relu activation, Global Avg Pool 1D, Batch Normalization and Dropout 0.2, 3 blocks of LSTM with 100 units, Batch Normalization and Dropout 0.3, 1 Dense with 3 units and softmax activation |

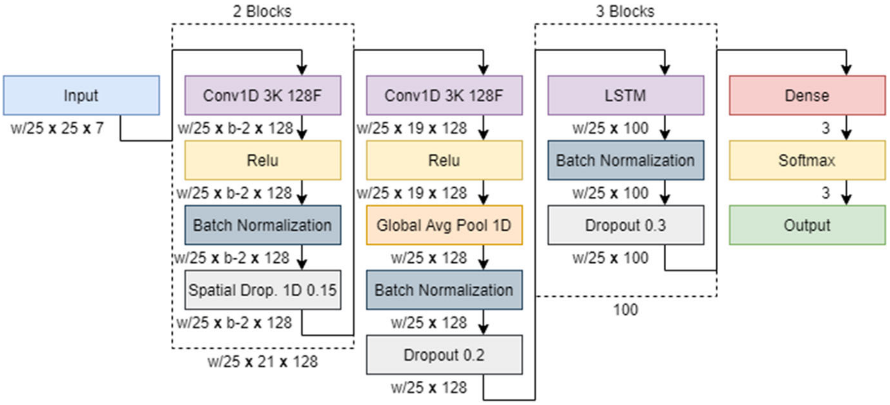


**Fig. 9** Best LSTM-CNN-based DNN model. *Source* the authors

**Table 8** Validation accuracy values for KMC

| Clusters | Window | | | | | Average |
|---|---|---|---|---|---|---|
| | 100 | 150 | 200 | 250 | 300 | |
| 3 | 56.20% | 58.13% | 59.10% | 59.67% | ***60.42%*** | 58.70% |

The values in bold and italics correspond to the emphasis on the highest accuracy value among all experiments and all models.

**Table 9** Validation accuracy values for SVM

| Kernel | Window | | | | | Average |
|---|---|---|---|---|---|---|
| | 100 | 150 | 200 | 250 | 300 | |
| Poly 3 | 65.00% | 66.60% | 68.05% | 68.07% | **68.71%** | 67.28% |
| rbf | 71.95% | 72.28% | ***72.68%*** | 72.42% | 72.45% | 72.36% |
| Sigmoid | 52.80% | 47.08% | **57.50%** | 54.73% | 56.75% | 53.77% |
| Average | 63.25% | 61.99% | 66.08% | 65.08% | 65.97% | 64.47% |

The values in bold correspond to the emphasis on the highest accuracy value among all the experiments of a given model. The values in bold and italics correspond to the emphasis on the highest accuracy value among all experiments and all models.

## 7 Results analysis

All models developed in this study were programmed in Python 3. The Classical Machine Learning models used the Scikit-Learn library, while the Deep Learning models used the Keras framework with Tensorflow backend. The experiments were performed on Google Collaboratory, with an NVIDIA Tesla P100 GPU with 12 GB and 25 GB RAM. In all techniques, except KNN, the experiments were performed 3 times in order to minimize the randomness of the initial parameters, recovering only the best among all three. The best model was considered to be the one that presented the highest accuracy value during the validation. The results obtained in the application of Classical Machine Learning are detailed in Tables 8, 9 and 10. The detailed value corresponds to the average accuracy in validation of the three experiments for a given data window size and the configuration of the hyperparameters of the technique.

For the KMC technique, the best resulting model for 3 clusters is the window with 300 samples, reaching 58.24% training accuracy and 60.42% validation accuracy. In this technique, the larger was the applied window, the greater were the accuracy values found. In the SVM technique, the sigmoid kernel obtained the worst accuracy values for all data windows, followed by the polynomial kernel with degree 3. Thus, the rbf kernel obtained the best accuracy values for all window variations, and the model with 200 samples obtained an accuracy of 76.15% for training and 72.68% for validation. In this kernel, the data window size has little influence on the final result. Finally, KNN presents better results with the number of neighbors between 1 and 50, where the window size also does not have much influence on the result. With large numbers of neighbors, the result tends to present a greater variation according to the number of samples. For the number of neighbors between 1 and 50, the window with 200

**Table 10** Validation accuracy values for KNN

| Neighbors | Window | | | | | Average |
|---|---|---|---|---|---|---|
| | 100 | 150 | 200 | 250 | 300 | |
| 1 | 72.28% | 72.77% | **74.46%** | 73.45% | 73.83% | 73.36% |
| 2 | 59.80% | 61.08% | **62.69%** | 60.87% | 61.92% | 61.27% |
| 5 | 72.37% | 72.78% | ***74.79%*** | 73.80% | 73.97% | 73.54% |
| 10 | 67.97% | 68.39% | **70.23%** | 68.81% | 69.72% | 69.02% |
| 50 | 70.83% | 71.54% | **72.01%** | 71.30% | 69.66% | 71.07% |
| 100 | 69.40% | **69.75%** | 69.47% | 68.77% | 67.63% | 69.00% |
| 250 | **66.25%** | 65.18% | 63.84% | 62.61% | 61.24% | 63.83% |
| 500 | **62.15%** | 59.57% | 57.66% | 54.65% | 53.93% | 57.59% |
| 1000 | **54.93%** | 51.42% | 49.12% | 45.83% | 44.05% | 49.07% |
| Average | 66.22% | 65.83% | 66.03% | 64.45% | 63.99% | 65.31% |

The values in bold correspond to the emphasis on the highest accuracy value among all the experiments of a given model. The values in bold and italics correspond to the emphasis on the highest accuracy value among all experiments and all models.

**Table 11** Accuracy values for each best model of classical machine learning techniques

| Model | Stage | Exp. 1 | Exp. 2 | Exp. 3 | Average |
|---|---|---|---|---|---|
| KMC | Training | 60.53% | 56.04% | 58.13% | 58.24% |
| | Validation | 65.67% | 60.32% | 55.26% | 60.42% |
| SVM | Training | 75.59% | 77.03% | 75.84% | 76.15% |
| | Validation | 67.28% | 75.61% | 75.16% | 72.68% |
| KNN | Training | 84.50% | 86.44% | 86.76% | 85.90% |
| | Validation | 76.28% | 71.11% | 76.98% | ***74.79%*** |

The values in bold and italics correspond to the emphasis on the highest accuracy value among all experiments and all models.

samples presents the best results, placing the best model with 5 neighbors, where it obtained accuracy in training dataset of 85.90% and 74.79% in the validation. With these results, the best models of each technique are detailed in the Table 11, separated by experiment.

Observing Table 11 results, we realize that the KNN technique not only has the highest accuracy average, but it is also the most stable technique, with the least variation in results between experiments with different contexts. The KMC presents a variance of 18.07%, the SVM of 14.63%, and the KNN of 6.85%. In addition, observing the confusion matrix of the best models in Fig. 10, KNN is the only technique in which the best model has more true positives and true negatives than false negatives and false positives, when considering each surface type separately. Therefore, as detailed in the Table 12, KNN has the best values for f1-score for two of the three data classes, classified dirt roads with f1-score of 65.39%, cobblestone roads with 53.65% and asphalt roads with 95.97%.

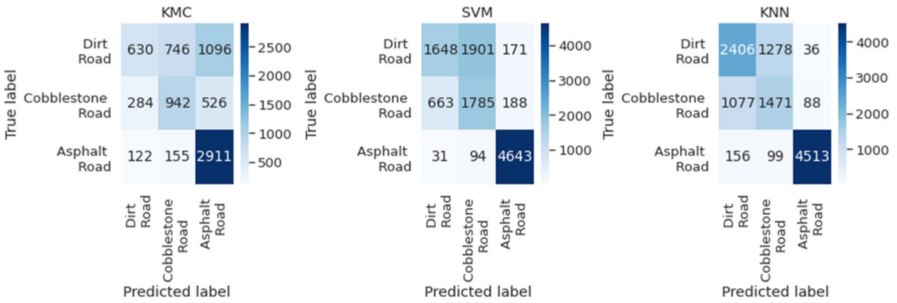


**Fig. 10** Confusion matrix for each best model of classical machine learning techniques. *Source* the authors

**Table 12** Evaluation metrics for each best model of classical machine learning techniques

| Model | Data class | F1 Score | Precision | Recall |
|---|---|---|---|---|
| KMC | Asphalt Road | 75.40% | 64.22% | 91.31% |
| | Cobblestone Road | 52.41% | 51.11% | 53.77% |
| | Dirt Road | 35.92% | 60.81% | 25.49% |
| SVM | Asphalt Road | 95.05% | 92.82% | 97.38% |
| | Cobblestone Road | 55.64% | 47.22% | 67.71% |
| | Dirt Road | 54.37% | 70.37% | 44.30% |
| KNN | Asphalt Road | 95.97% | 97.33% | 94.65% |
| | Cobblestone Road | 53.65% | 51.65% | 55.80% |
| | Dirt Road | 65.39% | 66.12% | 64.68% |

For experiments with Deep Learning techniques, the different methods for normalizing signals were initially evaluated. Among the three experimented, the Min Max Scaler between (0,1) presented worse results, reaching a higher loss value and a lower accuracy. The Robust Scaler was the second best, maintaining the signal in the data, but without scaling in an interval. Finally, the normalized data with Min Max Scaler between (−1, 1), maintaining the signal and scaling in a fixed interval, showed a faster convergence, less loss and greater accuracy. Thus, scalers that maintain the signal obtained better results, since the signal is important in this study, indicating direction information and not just a decrease in value. Using the normalized data, the proposed DNN models were tested. The results are detailed in Tables 13, 14 and 15.

All proposed DNN models, even in the worst results, have a higher accuracy value in training and validation than the best model based on Classical Machine Learning. This highlights the ability of Deep Learning to learn more complex relationships between data, compared to classical techniques. In all experiments, the impact on accuracy given the sample window size is practically insignificant, whereas classical techniques vary widely. In all experimented DNN approaches, the use of Bacth Normalization layer, in addition to speeding up training, has improved results in all scenarios. The use of Dropout layer also proved to be important for the model generalization. In LSTM-based models, the use of bidirectional layers practically did not change the results,

**Table 13** Validation accuracy values for LSTM-based DNNs

| Model | Window | | | | | Average |
|---|---|---|---|---|---|---|
| | 100 | 150 | 200 | 250 | 300 | |
| LSTM 1 | 89.14% | 89.44% | **89.86%** | 86.30% | 87.59% | 88.47% |
| LSTM 2 | **87.38%** | 87.33% | 86.80% | 86.26% | 85.85% | 86.72% |
| LSTM 3 | 89.11% | 89.61% | 89.99% | 89.21% | **90.45%** | 89.67% |
| LSTM 4 | 87.12% | **87.31%** | 87.07% | 86.11% | 86.48% | 86.82% |
| LSTM 5 | 89.74% | 89.14% | 90.15% | **90.60%** | 89.63% | 89.85% |
| LSTM 6 | 88.66% | 89.69% | **90.41%** | 88.99% | 90.88% | 89.72% |
| LSTM 7 | 90.39% | 91.37% | 91.85% | 92.71% | ***92.73%*** | 91.81% |
| **Average** | 88.79% | 89.13% | 89.45% | 88.60% | 89.09% | 89.01% |

The values in bold correspond to the emphasis on the highest accuracy value among all the experiments of a given model. The values in bold and italics correspond to the emphasis on the highest accuracy value among all experiments and all models.

**Table 14** Validation accuracy values for CNN-based DNNs

| Model | Window | | | | | Average |
|---|---|---|---|---|---|---|
| | 100 | 150 | 200 | 250 | 300 | |
| CNN 1 | 87.44% | 87.60% | **88.25%** | 87.07% | 87.54% | 87.58% |
| CNN 2 | 87.58% | 87.82% | 88.25% | **88.65%** | 87.94% | 88.05% |
| CNN 3 | 89.98% | 91.67% | 92.91% | 92.81% | **93.02%** | 92.08% |
| CNN 4 | 88.88% | 90.20% | 91.13% | 91.78% | **92.00%** | 90.80% |
| CNN 5 | 86.87% | 87.71% | **88.61%** | 87.81% | 88.35% | 87.87% |
| CNN 6 | 90.09% | 91.12% | 91.61% | **92.17%** | 91.94% | 91.39% |
| CNN 7 | 89.17% | 89.59% | **90.14%** | 89.66% | 89.65% | 89.64% |
| CNN 8 | 91.02% | 91.83% | 92.89% | 92.99% | ***93.17%*** | 92.38% |
| **Average** | 88.88% | 89.69% | 90.47% | 90.37% | 90.45% | 89.97% |

The values in bold correspond to the emphasis on the highest accuracy value among all the experiments of a given model. The values in bold and italics correspond to the emphasis on the highest accuracy value among all experiments and all models.

with an increase in accuracy for some windows, and a decrease for others, always in practically irrelevant values and, on average, less than 1%. The use of LSTM returned sequence vectors connected directly in the fully connected layers turned the results worse in all experiments, decreasing the accuracy value by up to 4%. Thus, the LSTM-based model with the best results is LSTM 7, in a window of 300 samples, where it reached a training accuracy value of 96.67% and 92.73% of validation accuracy.

In CNN-based networks, the use of two to four convolution layers proved to be essential for a good feature extraction. At the final output of the convolution blocks for connection to the fully connected layers block, the use of pooling improved the results when compared to the use of flattening only. Both max pooling and the global average pooling improved the results, with the second being significantly better. Thus,

**Table 15** Validation accuracy values for CNN-LSTM-based DNNs

| Model | Window | | | | | Average |
|---|---|---|---|---|---|---|
| | 100 | 150 | 200 | 250 | 300 | |
| CNN-LSTM 1 | 87.56% | 89.15% | 89.35% | **90.97%** | 90.78% | 89.56% |
| CNN-LSTM 2 | 87.92% | 88.38% | 89.76% | 89.54% | **90.47%** | 89.21% |
| CNN-LSTM 3 | 87.45% | 88.77% | 90.09% | 89.79% | **91.04%** | 89.43% |
| CNN-LSTM 4 | 87.43% | 89.43% | 89.51% | 89.93% | **90.99%** | 89.46% |
| CNN-LSTM 5 | 90.03% | 90.69% | 91.23% | 91.54% | **92.71%** | 91.24% |
| CNN-LSTM 6 | 90.27% | 91.50% | 92.10% | 92.48% | ***92.77%*** | 91.82% |
| **Average** | 88.44% | 89.65% | 90.34% | 90.71% | 91.46% | 90.12% |

The values in bold correspond to the emphasis on the highest accuracy value among all the experiments of a given model. The values in bold and italics correspond to the emphasis on the highest accuracy value among all experiments and all models.

**Table 16** Validation accuracy for each best model of deep machine learning techniques

| Model | Stage | Exp. 1 | Exp. 2 | Exp. 3 | Average |
|---|---|---|---|---|---|
| LSTM | Training | 98.54% | 98.45% | 93.03% | 96.67% |
| | Validation | 93.60% | 91.88% | 92.70% | 92.73% |
| CNN | Training | 94.39% | 97.62% | 94.67% | 95.56% |
| | Validation | 94.93% | 92.25% | 92.33% | ***93.17%*** |
| CNN-LSTM | Training | 91.68% | 98.85% | 94.16% | 94.90% |
| | Validation | 93.68% | 92.67% | 91.97% | 92.77% |

The values in bold and italics correspond to the emphasis on the highest accuracy value among all experiments and all models.

all experiments in which the average window accuracy was greater than 90%, there was the use of the global average pool. The CNN-based model with the best results is CNN 8, with a window of 300 samples, reaching training accuracy of 95.56% and 93.17% in the validation accuracy. Finally, on CNN-LSTM-based networks the same considerations as LSTM-based and CNN-based can be apply, with the best model being CNN-LSTM 6 with a window of 300 samples, obtaining training accuracy of 94.9% and 92.77% validation accuracy. The best DNN models for each approach are detailed in Table 16.

Based on the Table 16, we observe that the different approaches of Deep Learning proved to be suitable for the road surface type classification problem. Evaluating the average validation accuracy, the CNN-based model was slightly better than the others, although the LSTM-based was better in experiment 3, the CNN-based was the best in experiment 1, and CNN-LSTM-based the best in experiment 2. Analyzing the confusion matrix in the Fig. 11 and other evaluation metrics detailed in Table 17, we observed that the CNN-based model has the highest f1-score values for all data classes, being the best model when considering not only its ability to hit true positives and true negatives, but also to avoid false positives and false negatives. Therefore, we consider CNN-based the best model based on Deep Learning for the road surface type

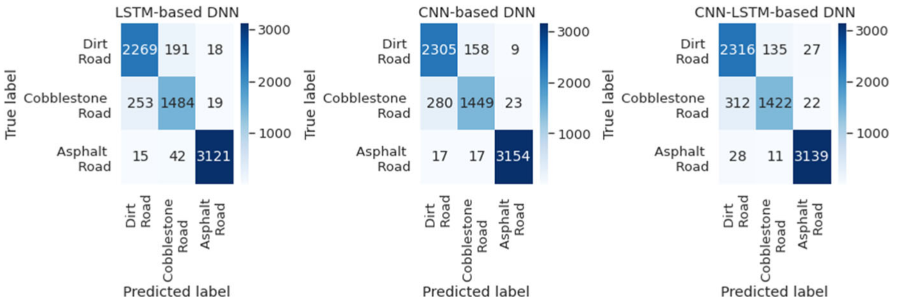


**Fig. 11** Confusion matrix for each best model of deep learning techniques. *Source* the authors

**Table 17** Evaluation metrics for each best model of deep machine learning techniques

| Model | Data class | F1 score | Precision | Recall |
|---|---|---|---|---|
| LSTM | Asphalt Road | 98.52% | 98.83% | 98.21% |
| | Cobblestone Road | 85.46% | 86.43% | 84.51% |
| | Dirt Road | 90.49% | 89.44% | 91.57% |
| CNN | Asphalt Road | 98.96% | 99.00% | 98.93% |
| | Cobblestone Road | 85.84% | 89.23% | 82.71% |
| | Dirt Road | 90.85% | 88.59% | 93.24% |
| CNN-LSTM | Asphalt Road | 98.62% | 98.46% | 98.77% |
| | Cobblestone Road | 85.56% | 90.69% | 80.98% |
| | Dirt Road | 90.22% | 87.20% | 93.46% |

classification due to its greater accuracy and f1-score values, in addition to being the model with the lowest computational cost among the three DNN models. Thus, the technique obtained an accuracy value in validation of 93.17%, classifying dirt road with f1-score value of 90.85%, cobblestone road with 85.84% and asphalt road with 98.96%, obtaining better results than classical methods and related studies, and in contextual variations.

## 8 Discussion and conclusions

With the ITS development, some applications have become increasily present in daily life, such as ADAS and, soon, autonomous vehicles. However, these systems need many sources of situational data to perform their purpose well. Among the various information, the recognition of surface type is one of the most important, and it can be used in the most varied applications to support human or machine decision making. However, for a wide application, it is necessary to develop a recognition model that uses safe technologies, and that is able to understand different contexts through which a vehicle can travel.

Based on the premise above, in this work we developed models for road surface type classification using inertial sensor signals, which by their passive approach are safe, non-polluting and generally low cost. Through the literature analysis, there are few related studies which were carried out with low accuracy values, without considering contextual variations. As a way to analyze and validate the model for application in real-world scenarios, we list the dependency factors that impact in the learning and generalization of the pattern recognition model, and aggregate them. Initially, we performed several data collections to obtain variations of these factors. After pre-processing these data, they were interpreted using Classical Machine Learning and Deep Learning techniques to identify which one was the most appropriate.

Among the Classical Machine Learning techniques, we apply KMC, SVM and KNN that obtained low accuracy values in the validation, 60.42%, 72.68% and 74.79%, respectively. In the application of these methods, we observed the difficulty of extracting high-level features to well represent the data classes, even when applied statistical methods which are widely adopted in the feature extraction of vibration-based data. In the application of Deep Learning approach, all techniques were able to extract complex features from the data, so in all the developed DNN models, accuracy values were greater than those obtained through the best developed model based on classical techniques. LSTM-based, CNN-based and CNN-LSTM-based models were analyzed, which obtained the best validation accuracy result of 92.73%, 93.17% and 92.77%, respectively. In a general analysis, we consider the CNN-based model the best due to the f1-score and accuracy value. The model classified segments of dirt road with f1-score value of 90.85%, cobblestone road with 85.84% and asphalt road with 98.96%. With these results, we notice that all the data classes were properly handled by the model, and it is not necessary to apply data class balancing. In this work, we did not compare our models with those related in the state-of-the-art due to the lack of detail on the methodology of related studies.

With the public availability of the datasets that we produce, new studies can be carried out on the recognition and classification of the most diverse patterns in ITS. Related to the exteroception, can be recognized potholes, speed bumps, surface quality, among others. Related to proprioception, can be recognized driving events such as acceleration, braking, turning right or left, as well as driving safety level. In relation to the road surface type classification, new researches can perform experiments with the permutation of datasets grouped by dependency factors, analyze the other data collection points, such as near and above the suspension, and on the dashboard, both present in the collected dataset. Deep Learning layers with different approaches can also be analyzed, such as the Gated Recurrent Unit (GRU) and Convolutional LSTM (ConvLSTM).

## Compliance with ethical standards

**Conflict of interest** The authors declare that they have no confict of interest.

## References

1. Ahmad W, Kazmi BM, Ali H (2019) Human activity recognition using multi-head CNN followed by lstm. In: 2019 15th international conference on emerging technologies (ICET), pp 1–6

2. Al-Yafeai D, Darabseh T, Mourad AI (2019) Quarter vs. half car model energy harvesting systems. In: 2019 Advances in science and engineering technology international conferences (ASET), pp 1–5. https://doi.org/10.1109/ICASET.2019.8714433
3. Alemayoh TT, Hoon Lee J, Okamoto S (2019) Deep learning based real-time daily human activity recognition and its implementation in a smartphone. In: 2019 16th international conference on ubiquitous robots (UR), pp 179–182
4. Alqudah YA, Sababha BH (2016) A statistical approach to estimating driving events by a smartphone. In: 2016 international conference on computational science and computational intelligence (CSCI), pp 1021–1025. https://doi.org/10.1109/CSCI.2016.0195
5. Andria G, Attivissimo F, Nisio AD, Lanzolla A, Pellegrino A (2016) Development of an automotive data acquisition platform for analysis of driving behavior. Measurement 93:278–287. https://doi.org/10.1016/j.measurement.2016.07.035
6. Bello-Salau H, Aibinu A, Onumanyi A, Onwuka E, Dukiya J, Ohize H (2018) New road anomaly detection and characterization algorithm for autonomous vehicles. Appl Comput Inform. https://doi.org/10.1016/j.aci.2018.05.002
7. Bose B, Dutta J, Ghosh S, Pramanick P, Roy S (2018) Smartphone based system for real-time aggressive driving detection and marking rash driving-prone areas. In: Proceedings of the workshop program of the 19th international conference on distributed computing and networking, workshops ICDCN ’18, pp 27:1–27:6. ACM, New York, NY, USA. https://doi.org/10.1145/3170521.3170549
8. Braga N (2017) Manual de Mecatrônica. Editora NCB. https://books.google.com.br/books?id=Zr3iBgAAQBAJ
9. Brownlee J (2020) Imbalanced classification with python: better metrics, balance skewed classes, cost-sensitive learning. Machine Learning Mastery https://books.google.com.br/books?id=jaXJDwAAQBAJ
10. Brunauer R, Rehrl K (2016) Supporting road maintenance with in-vehicle data: Results from a field trial on road surface condition monitoring. In: 2016 IEEE 19th international conference on intelligent transportation systems (ITSC), pp 2236–2241. https://doi.org/10.1109/ITSC.2016.7795917
11. Chen Y, Xue Y (2015) A deep learning approach to human activity recognition based on single accelerometer. In: 2015 IEEE international conference on systems, man, and cybernetics, pp 1488–1492
12. Deep S, Zheng X (2019) Hybrid model featuring CNN and LSTM architecture for human activity recognition on smartphone sensor data. In: 2019 20th international conference on parallel and distributed computing, applications and technologies (PDCAT), pp 259–264
13. Dixon P, Schütte K, Vanwanseele B, Jacobs J, Dennerlein J, Schiffman J, Fournier PA, Hu B (2019) Machine learning algorithms can classify outdoor terrain types during running using accelerometry data. Gait and Posture 74:176–181. https://doi.org/10.1016/j.gaitpost.2019.09.005
14. Douangphachanh V, Oneyama H (2013) Estimation of road roughness condition from smartphones under realistic settings. In: 2013 13th international conference on ITS telecommunications (ITST), pp 433–439. https://doi.org/10.1109/ITST.2013.6685585
15. Fernández A, García S, Galar M, Prati RC, Krawczyk B, Herrera F (2018) Learning from imbalanced data sets. Springer, Berlin
16. Foley D (2019) K-means clustering. https://towardsdatascience.com/k-means-clustering-8e1e64c1561c
17. Goodfellow I, Bengio Y, Courville A (2016) Deep learning. Adaptive computation and machine learning series. MIT Press, London. https://books.google.com.br/books?id=Np9SDQAAQBAJ
18. Groves P (2013) Principles of GNSS, inertial, and multisensor integrated navigation systems, 2nd edn. GNSS/GPS. Artech House. https://books.google.com.br/books?id=t94fAgAAQBAJ
19. Gueta LB, Sato A (2017) Classifying road surface conditions using vibration signals. In: 2017 Asia-Pacific signal and information processing association annual summit and conference (APSIPA ASC), pp 039–043. https://doi.org/10.1109/APSIPA.2017.8281999
20. He H, Ma Y (2013) Imbalanced learning: foundations, algorithms, and applications. Wiley, London
21. Hou Y, Gupta A, Guan T, Hu S, Su L, Qiao C (2017) Vehsense: slippery road detection using smartphones. In: 2017 IEEE 85th vehicular technology conference (VTC Spring), pp 1–5. https://doi.org/10.1109/VTCSpring.2017.8108301
22. Jones MT (2017) Arquiteturas de aprendizado profundo: O surgimento da inteligência artificial. https://www.ibm.com/developerworks/br/library/cc-machine-learning-deep-learning-architectures/index.html

23. Khaleghian S, Taheri S (2017) Terrain classification using intelligent tire. J Terramech 71:15–24. https://doi.org/10.1016/j.jterra.2017.01.005
24. Khandelwal R (2018) K-nearest neighbors (KNN). https://medium.com/datadriveninvestor/k-nearest-neighbors-knn-7b4bd0128da7
25. Krawczyk B (2016) Learning from imbalanced data: open challenges and future directions. Progr Artif Intell 5(4):221–232
26. Kuhn M, Johnson K et al (2013) Applied predictive modeling, vol 26. Springer, Berlin
27. Kumar GA, Kumar AS, Kumar AA, Maharajothi T (2017) Road quality management system using mobile sensors. In: 2017 international conference on innovations in information, embedded and communication systems (ICIIECS), pp 1–6. https://doi.org/10.1109/ICIIECS.2017.8276014
28. LeCun Y, Bengio Y, Hinton G (2015) Deep learning. Nature 521(7553):436
29. Li F, Zhang H, Che H, Qiu X (2016) Dangerous driving behavior detection using smartphone sensors. In: 2016 IEEE 19th international conference on intelligent transportation systems (ITSC), pp 1902–1907. https://doi.org/10.1109/ITSC.2016.7795864
30. Lima LC, Amorim VJP, Pereira IM, Ribeiro FN, Oliveira RAR (2016) Using crowdsourcing techniques and mobile devices for asphaltic pavement quality recognition. In: 2016 VI Brazilian symposium on computing systems engineering (SBESC), pp 144–149. https://doi.org/10.1109/SBESC.2016.029
31. Gopi VP (2017) Vehicle vibration signal processing for road surface monitoring. IEEE Sens J 17(16):5192–5197. https://doi.org/10.1109/JSEN.2017.2719865
32. Machin M, Sanguesa JA, Garrido P, Martinez FJ (2018) On the use of artificial intelligence techniques in intelligent transportation systems. In: 2018 IEEE wireless communications and networking conference workshops (WCNCW), pp 332–337
33. Mathew TV (2014) Transportation systems engineering: Intrusive technologies. https://nptel.ac.in/courses/105101008/downloads/cete_09.pdf
34. Mathew TV (2014) Transportation systems engineering: Non-intrusive technologies. https://nptel.ac.in/courses/105101008/downloads/cete_10.pdf
35. Menegazzo J, Von Wangenheim A (2018) Vehicular perception and proprioception based on inertial sensing: a systematic review. Technical report, Federal University of Santa Catarina—Brazilian Institute for Digital Convergence. https://doi.org/10.13140/RG.2.2.20113.07525
36. Menegazzo J, von Wangenheim A (2020) Vehicular perception based on inertial sensing: a structured mapping of approaches and methods. SN Comput Sci. https://doi.org/10.1007/s42979-020-00275-z
37. Nalavde R, Surve A (2015) Driver assistant services using ubiquitous smartphone. In: 2015 international conference on communications and signal processing (ICCSP), pp 0430–0434. https://doi.org/10.1109/ICCSP.2015.7322924
38. Nisbet R, Elder J, Miner G (2009) Handbook of statistical analysis and data mining applications. Elsevier, Amsterdam
39. Pholprasit T, Choochaiwattana W, Saiprasert C (2015) A comparison of driving behaviour prediction algorithm using multi-sensory data on a smartphone. In: 2015 IEEE/ACIs 16th international conference on software engineering, artificial intelligence, networking and parallel/distributed computing (SNPD), pp 1–6 . https://doi.org/10.1109/SNPD.2015.7176249
40. Prapulla SB, Rao SN, Herur VA (2017) Road quality analysis and mapping for faster and safer travel. In: 2017 international conference on energy, communication, data analytics and soft computing (ICECDS), pp 2487–2490. https://doi.org/10.1109/ICECDS.2017.8389899
41. Rateke T, Justen KA, von Wangenheim A (2019) Road surface classification with images captured from low-cost camera-road traversing knowledge (RTK) dataset. Rev Inform Teórica e Apl 26(3):50–64
42. Savera A, Zia A, Edhi MS, Tauseen M, Shamsi JA (2016) Bumpster: A mobile cloud computing system for speed breakers and ditches. In: 2016 IEEE 41st conference on local computer networks workshops (LCN Workshops), pp 65–71. https://doi.org/10.1109/LCN.2016.030
43. Sebastian B, Ben-Tzvi P (2019) Support vector machine based real-time terrain estimation for tracked robots. Mechatronics 62:102260. https://doi.org/10.1016/j.mechatronics.2019.102260
44. Shrestha A, Won M (2018) Deepwalking: enabling smartphone-based walking speed estimation using deep learning. In: 2018 IEEE global communications conference (GLOBECOM), pp 1–6
45. Shubham J (2018) Support vector machines (SVM). https://medium.com/coinmonks/support-vector-machines-svm-b2b433419d73
46. Singh B, Gupta A (2015) Recent trends in intelligent transportation systems: a review. J Transp Lit 9:30–34

47. Singh G, Bansal D, Sofat S, Aggarwal N (2017) Smart patrolling: an efficient road surface monitoring using smartphone sensors and crowdsourcing. Pervas Mobile Comput 40:71–88. https://doi.org/10.1016/j.pmcj.2017.06.002
48. Souza VM (2018) Asphalt pavement classification using smartphone accelerometer and complexity invariant distance. Eng Appl Artif Intell 74:198–211. https://doi.org/10.1016/j.engappai.2018.06.003
49. Tolentino-Rabelo R, Muñoz DM (2016) Online terrain classification for mobile robots using FPGAS. In: 2016 IEEE 7th Latin American symposium on circuits systems (LASCAS), pp 231–234
50. Wang J, Chen Y, Hao S, Peng X, Hu L (2019) Deep learning for sensor-based activity recognition: a survey. Pattern Recogn Lett 119:3–11. https://doi.org/10.1016/j.patrec.2018.02.010
51. Wang S, Kodagoda S, Shi L, Dai X (2018) Two-stage road terrain identification approach for land vehicles using feature-based and Markov random field algorithm. IEEE Intell Syst 33(1):29–39
52. Wang S, Kodagoda S, Shi L, Wang H (2017) Road-terrain classification for land vehicles: employing an acceleration-based approach. IEEE Veh Technol Mag 12(3):34–41
53. Wickramarathne T, Garg V, Bauer P (2018) On the use of 3-d accelerometers for road quality assessment. In: 2018 IEEE 87th vehicular technology conference (VTC Spring), pp 1–5. https://doi.org/10.1109/VTCSpring.2018.8417779
54. Yang J, Cheng K, Chen J, Zhou B, Li Q (2018) Smartphones based online activity recognition for indoor localization using deep convolutional neural network. In: 2018 ubiquitous positioning, indoor navigation and location-based services (UPINLBS), pp 1–7
55. Zaccone G, Karim M, Menshawy A (2017) Deep learning with TensorFlow. Packt Publishing. https://books.google.com.br/books?id=C0IwDwAAQBAJ
56. Zebin T, Balaban E, Ozanyan KB, Casson AJ, Peek N (2019) Implementation of a batch normalized deep lstm recurrent network on a smartphone for human activity recognition. In: 2019 IEEE EMBS international conference on biomedical health informatics (BHI), pp 1–4
57. Zebin T, Sperrin M, Peek N, Casson AJ (2018) Human activity recognition from inertial sensor time-series using batch normalized deep lstm recurrent networks. In: 2018 40th annual international conference of the IEEE engineering in medicine and biology society (EMBC), pp 1–4

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

## Affiliations

### Jeferson Menegazzo[1] · Aldo von Wangenheim[1]

✉ Jeferson Menegazzo
jeferson.menegazzo@posgrad.ufsc.br

Aldo von Wangenheim
aldo.vw@ufsc.br

[1] Graduate Program in Computer Science (PPGCC), Image Processing and Computer Graphics Lab (LAPIX), Brazilian National Institute for Digital Convergence (INCoD), Department of Informatics and Statistics (INE), Federal University of Santa Catarina (UFSC), Florianópolis, Santa Catarina, Brazil