Low-Power Real-Time TinyML Approach for Pothole and Speed Bump Classification Using IMU Data on FPGA

César Melo
Bosch Car Multimedia
Braga. 4705, Portugal
cesar.melo@pt.bosch.com

Pedro Delgado
Bosch Car Multimedia
Braga. 4705, Portugal
pedro.delgado@pt.bosch.com

Gabriel Carvalho
Bosch Car Multimedia
Braga. 4705, Portugal
gabriel.carvalho2@pt.bosch.com

Rui Silva
LASI/ALGORITMI
University of Minho
Braga, 4705, Portugal
rsilva@algoritmi.uminho.pt

Hugo Peixoto
LASI/ALGORITMI
University of Minho
Braga, 4705, Portugal
hpeixoto@di.uminho.pt

José Machado
LASI/ALGORITMI
University of Minho
Braga, 4705, Portugal
jmac@di.uminho.pt

2025 Global Congress on Emerging Technologies (GCET) | 979-8-3315-7178-8/25/$31.00 ©2025 IEEE | DOI: 10.1109/GCET68529.2025.11450715

***Abstract* - Monitoring the condition of road surfaces can bring benefits to drivers, ranging from increased safety and comfort, to cost reduction. In this context, accelerometer-based approaches offer a cheap and robust solution to the mitigation of common issues that arise from other types of sensors, like cameras, namely computational cost concerns and inability in low visibility scenarios. Nonetheless, as more applications compete for resources in modern vehicles, it becomes imperative to design such applications considering power efficiency constraints (TinyML): to this end, we propose a lightweight machine learning model to detect road irregularities (speed bumps, potholes, and manholes, including the side of impact) using 4 Inertial Measurement Unit (IMU) mounted near each vehicle wheel. Furthermore, to have a real-time monitoring system, we deploy an XGBoost model on a Field-Programmable Gate Array (FPGA) through the use of the Conifer framework, to translate the model using High Level Synthesis (HLS) language. Our solution achieves an accuracy of 88.1%, with 750 microseconds of inference time, showcasing a robust performance in varied scenarios.**

Keywords – Pothole Detection, TinyML, Accelerometers, Conifer, Machine Learning, Automotive Application

I. INTRODUCTION

Road surface condition can significantly impact multiple factors of driving, such as safety, comfort, vehicle wear and fuel efficiency. Repeateadly hitting potholes deteriorates the vehicle's suspension system, causing misalignment and damage on the tires, which can lead to punctures and accidents in extreme cases. A survey by the American Automobile Association estimated that in 2021, potholes caused $26.5 billion in vehicle repairs in the United States, with as many as 10% of drivers needing a repair after hitting a pothole [1]. Speed bumps also create discomfort and negatively impact fuel efficiency due to the constant need of breaking and accelerating [2].

Given the evident need of road condition monitoring, many recent solutions have been proposed, most of which leveraging vehicles themselves to detect road defects automatically. While these solutions often employ optical sensors, such as cameras, the lower cost of Inertial Measurement Units (IMUs) allied with their robustness to low visibility scenarios makes them a popular choice for such applications. Recent developments in Advanced Driver-Assistance Systems (ADAS) lead to increasingly more computation resources being needed in modern vehicles, creating power consumption concerns especially in electric vehicles. The usage of IMUs instead of cameras presents an improvement in power consumption due to the comparatively smaller ammount of generated data, which results in fewer computational resources.

Hence, it is important that the sensor data processing stage is designed with power efficiency in mind, and we propose a lightweight pothole and speed bump detection machine learning model based on IMU data, capable of distinguishing the pothole's impact side. This feature is valuable to give the drivers a report information about potential damage on the vehicle, whether it is related with the tires or suspension. The model's design process took into consideration multiple factors, such as the most valuable features extracted from raw IMU signals, different data acquisition rates, and the amount of data required to perform the predictions. Furthermore, we explore the Conifer framework [3] to automatically deploy our classification model on a Field-Programmable Gate Array (FPGA). This work can be used as a reference for several different research areas that include IMUs and FPGAs, since we describe how to create and fully optimize an entire scalable pipeline, from data collection to deployment, giving an overview of how to process the data, perform and optimize the feature selection stage, and how to tackle constraints in relevant topics such as memory, latency, and power consumption.

II. STATE OF THE ART REVIEW

A. Pothole and Speed Bump Detection

Choosing FPGA as the target hardware for the Machine Learning (ML) model deployment has gained popularity over the past years, as the fast time-to-market, flexibility, prototyping and high performance are combined with low latency and power consumption. The road condition monitoring studies using FPGA mostly focus on computer vision approaches by combining image data with traditional Neural Networks [4, 5] and YOLO object detectors [6, 7]. Even when targetting other platforms, like microcontrollers, similar approaches are used [8, 9]. However, these techniques present limitations in low visibility scenarios, with vibration-based methods overcoming these constraints by using different types of sensors, such as IMUs. The most standard approaches simply compare statistical information from the Z acceleration axis, like amplitude, difference between

Authorized licensed use limited to: b-on: UNIVERSIDADE DO MINHO. Downloaded on April 23,2026 at 15:01:09 UTC from IEEE Xplore. Restrictions apply.

consecutive measurements, and standard deviation, against predefined thresholds [10, 11]. However, the threshold values are static, susceptible to noise, and empirically defined according to the data collected at the time of experiments.

More recent works focus on the extraction of complex features from accelerometers in combination with traditional machine learning models. These features can be time-based [12], frequency-based [13], and wavelet-based [14]. Although the combination of different types of features can enhance the model's performance, often this has a negative impact on other important metrics for automotive applications, such as extraction time and power consumption. Regarding time-based features, most of the architectures used in the abovementioned works were tested in conjuction with gyroscope data [15]. Out of 5 different classifiers, the Random Forest (RF) model performed the best with 94.44% accuracy. Despite being a similar pipeline when compared with our approach, it consists on a binary classification problem without considering the detection of speed bumps. In terms of model parameters, with the values of 311 and 46 for the number of estimators and max depth hyperparameters, respectively, the presented model is computationally much more expensive that the one we propose. Moreover, we go beyond the model's generation, with the focus being on the application of strategies that reduce the model's size and latency, targetting a complete solution from the data acquisition stage to the model's deployment in lower-power hardware.

B. *Conifer*

Conifer is an extension of the hls4ml framework [16] that specializes in translating Boosted Decision Trees (BDTs) to FPGA hardware description languages for extreme low latency inference: it supports popular BDT training frameworks including eXtreme Gradient Boosting (XGBoost) [17], being able to convert it to Vivado High Level Synthesis (HLS) and Verilog Hardware Description Language (HDL). XGBoost is a supervised tree-based machine learning model designed to perform very fast processing of large ammounts of tabular data on a parallel fashion. This model has two branches: ensembler, which combines the outputs from multiple decision trees to deliver a strong merged prediction; and booster, where each isolated decision tree is trained with the errors of the previous ones. The model's thresholds and scores are stored in fixed precision, being possible to define the bit width of the representation to maintain accuracy and, at the same time, reduce energy and the memory required to perform inference [18].

Showcasing Conifer capabilities, in this work a BDT was deployed on a Xilinx vu9p-flgb2104-2L-e FPGA with a target clock frequency of 200 MHz. All thresholds and scores were encoded with 18 bits, with the hardware implementation achieving very low latencies (60ns) and maintaining the accuracy of the original BDT.

III. METHODOLOGY

A. *Sensor Setup and Data Collection*

IMUs with 6-axis were placed near each of the 4 wheels above the suspension of a Mercedes E300d to better detect the acceleration and rotations arising from road irregularities, as in theory these are the most valuable positions to detect the impact side of the holes (please see Fig. 1) [19]. The data collection occurred in a total of 31 locations, with each location containing one of the following road irregularities: pothole, round manhole, rectangular manhole, and speed bump, driving at speeds from 10 to 80 km/h. For the sake of simplicity and availability, only asphalt roads were considered, and all the hole-related samples were merged into a single class named "Pothole". Not all locations and irregularity types were driven at the entire speed interval, due to limitations of the irregulatirities themselves or the speed limits of the roads they were located at.

Each recording consisted in driving over the irregularity at an approximately constant speed and, when applicable, with both left and right tires – by aligning the desired vehicle side with the hole, the movement started with a certain distance from the irregularity, far enough to accelerate until the desired speed. As to ensure that the tire had an impact in the pothole and constituted a valid sample, an external camera was used to film a slowed down version of the driving. Multiple recordings were made in each location varying speed, side of impact, and the suspension mode, which was made more or less rigid to mitigate result dependency regarding inherent vehicle characteristics.

Labelling consisted in defining the boundaries associated with crossing a pothole or speedbump in the IMU signals: to do so, a bottom/front-faced 360º camera was installed underneath the vehicle, with its wide field of view allowing the visualization of the road in front of the tires. Furthermore, a LED strip was also mounted right above the camera in order to synchronize the video recording with the data acquired. Some of the collected holes and speed bumps, alongside the camera and LED strip mounting location are depicted in Fig. 2, and Fig. 3, respectively. With reference to the camera's frame rate, the event timestamps are calculated and extrapolated to the collected signals – since the IMU's sampling rate (470Hz) is higher, a margin of 0.5 seconds is given before and after the events occur.

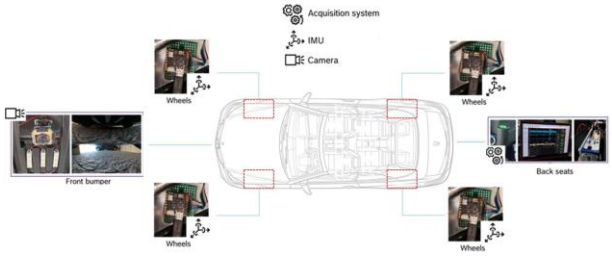


Fig. 1. IMU sensor placement in the test car. Adapted from [20].



Fig. 2. Potholes and speed bumps samples collected.

Authorized licensed use limited to: b-on: UNIVERSIDADE DO MINHO. Downloaded on April 23,2026 at 15:01:09 UTC from IEEE Xplore. Restrictions apply.



Fig. 3. Camera's field of view, with reference to the frame before (left) and after (right) the acquisition started.

B. Feature Extraction and Dataset Generation

Given several time-series sequences, the extraction of multiple time-based features will allow the classifiers to learn and discriminate the different classes, based on the values of those features. Table I describes the extracted features from the collected data.

The dataset generation followed a sliding window procedure to each recording, where an entire sequence is splitted into multiple fixed-size windows. Since each recording has a total of 24 raw signals (6 signals collected from 4 different IMUs), the feature extraction results in a total of 408 features for each window generated from the sliding window step – because the window size is based on the recording's time duration, the number of points that constitutes a window is directly dependent on the sensor's sampling rate.

Regardless of the window size, the label assignment for each sample is made by checking if the window's current timestamps are inside the defined ground-truth event boundaries defined for each recording. Taking the original sampling rate value of 470Hz as a starting point, further experiments with ML algorithms allowed the study of the optimal dataset configuration and model that results in the best performance in terms of accuracy, power consumption, and latency, with attention to the dataset's window length value, sensor's sampling rate value, model's hyperparameters, and dataset's number of features.

C. Classification Model

Considering Conifer for the model's deployment, XGBoost was the selected algorithm to perform the experiments due to the seemless integration with the aforementioned framework. Ultimately, it is desired that the algorithm predicts 1 out of 4 classes: "No Event", which means no irregularity was detected; "Pothole Right", when a hole was detected in the right side of the car; "Pothole Left", when a hole was detected in the left side of the car; and "Speed Bump" – to compare the performance across experiments, the accuracy metric is used by measuring how well the model correctly identifies the classes.

Regarding the dataset split, and considering each class in separate to maintain representability, 70% of the samples are allocated for training, 20% for validation, and 10% for test. As a way to mitigate the class imbalance resulting from the sliding window approach, the "No Event" samples are randomly cut to the same number of the second highest class. Table II describes the XGBoost's hyperparameters with the default values at the time of instantiating the model, that will be used until the stage of hyperparameter tuning.

TABLE I. FEATURES EXTRACTED FROM THE RAW IMU SIGNALS.

Feature	Description
Mean	Average of the values
Mean Absolute Difference	The average rate of absolute change of consecutive samples
Standard Deviation	Standard deviation of the signal values
Root Mean Square	Square-root of the average of the squared values
Amplitude Range	Difference between maximum and minimum values
Max	Maximum value of the signal
Min	Minimum value of the signal
Skewness	Asymmetry of the values distribution
Kurtosis	Tailedness of the values distribution
Negative Turning Points	Number of times that the signal goes from a higher to a lower value
Positive Turning Points	Number of times that the signal goes from a lower to a higher value
Zero Crossing Rate	Number of times that a signal's value is zero
Autocorrelation Lags 1-5	Autocorrelation of a signal with its 1-5 sample delay.

TABLE II. DEFAULT XGB'S HYPERPARAMETER VALUES.

Hyperparameters	Description	Default Value	GridSearch Values
Number of estimators	Number of trees to build during training	100	[50, 100, 200]
Learning Rate (η)	Importance of each individual tree on the final prediction	0.3	[0.2, 0.3, 0.5]
Max Depth	Depth of each individual tree	6	[3, 6, 9]

Fig. 4 presents the accuracies for the different performed experiments: in red, variation of the window length; in green, variation of the sensor's sampling rate; in blue, variation of the number of used features. Multiple datasets were created with different sizes (1, 2, 4, and 5 seconds) to find the optimal dataset's window length value, with an overlap of 0.5 seconds between windows. The red bars show that using a value of 5 seconds leads to the best performance (88.5% accuracy), as this is justified by the fact that the features are calculated from samples that have more information and temporal context.

Operational Competitiveness and Internationalization Programme (COMPETE 2020) / FCT – Fundação para a Ciência e Tecnologia

Authorized licensed use limited to: b-on: UNIVERSIDADE DO MINHO. Downloaded on April 23,2026 at 15:01:09 UTC from IEEE Xplore. Restrictions apply.

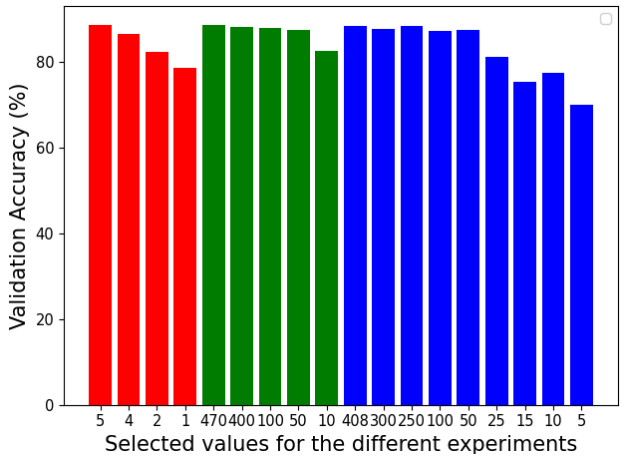


Fig. 4. Model's performance by: window length in seconds (red); sampling rate in Hz (green); number of features (blue).

Reducing sampling rate decreases the number of points used to calculate the features, leading to lower latency, memory, and energy consumptions; on the other hand, the model's ability to capture high frequency artifacts related with sudden changes in the data may be affected. Before downsampling the data to different rates, a low pass filter with 40% of the sampling frequency was applied as the cutoff to avoid an aliasing effect. Looking at the green bars, a very competitive performance is seen from the original value to 50Hz. The selected value was 100Hz (87.71% accuracy) since it is the best trade-off between accuracy and sampling rate – the degradation was more than two times higher when the sampling rate decreased from 100Hz to 50Hz (0.45%), than when it decreased from 400Hz to 100Hz (0.23%).

The number of used features is also relevant to speed up inference and reduce the computational requirements. This study was conducted on the model that achieved the best performance from the hyperparameter tuning step. The tweaked values for each hyperparameter are described on the last column of Table 2 – by testing all the combinations, the optimal configuration is [50, 0.3, 3] with a validation accuracy of 88.16%. The difference in 0.56% to the best performing configuration does not outweigh the faster inference time with the usage of 4 times less number of estimators.

Thus, the feature selection step consists of two iterative phases: first, based on the importance calculated from the impurity caused by each feature when it is used to split the samples at individual tree nodes; second, based on the removal of all the features related with a specific type. Maintaining the same criteria of finding the optimal value and with attention to the blue bars, the selected value is 50 (87.26% accuracy). Despite being slightly worse than other subsets, it results in a substantially lower feature extraction time. The following feature types were completely discarded after the first selection phase: Standard Deviation, Root Mean Square, Positive Turning Points, and Zero Crossing Rate.

After selecting the 50 most important features from the previous step, the next stage consists on evaluating the model's performance with the combination of specific feature types, which will help on the reduction of the mathematical operations associated to each type at the time of deploying the pipeline on the target hardware. Table III depicts the set of features that leads to the best model performance.

TABLE III. SENSOR SIGNALS AND CORRESPONDING FINAL FEATURES.

Sensor Location	Signal	Features
Front Right Wheel	ACC X	Max
	RATE X	Max, Min
	RATE Y	Autocorr Lag1, Kurtosis, Min
Front Left Wheel	ACC Z	Kurtosis
	RATE X	Min
	RATE Z	Max
Rear Right Wheel	ACC X	Kurtosis, Max
	ACC Y	Min
	ACC Z	Autocorr Lag1, Min
	RATE X	Max
	RATE Y	Autocorr Lag1
Rear Left Wheel	ACC X	Kurtosis, Max
	ACC Z	Autocorr Lag1
	RATE X	Min, Max
	RATE Y	Min

With a degradation of 0.56% in comparison with the subset of 50 features resulting from the previous step, the total total number of features was reduced from 50 to 22, with only 4 types still being used: Max, Min, Kurtosis, and Autocorrelation Lag1 – this feature set obtained a test set accuracy of 87.98%.

D. System Architecture and Model Deployment

A generic overview of the real-time system, from the stage of data collection up to the visualization results, can be given by a simple sequence (please see Fig. 5):

- The IMU data from each sensor is acquired by the FPGA via programmable logic (PL) through the Serial Peripheral Interface (SPI).
- The raw data acquired is sent to the FPGA's Processing System (PS) via Advanced eXtensible Interface (AXI) protocol.
- The features calculation step is performed using the required Python libraries.
- The feature values are sent to the PL to compute the inference on the deployed model with Conifer.
- The output is sent back to the FPGA's PS and sent via Universal Asynchronous Receiver/Transmitter (UART) to a graphical interface where the real-time results can be observed.

The first step after generating the model is the implementation of optimization techniques that can speed-up inference while reducing both memory usage and energy consumption, all while keeping the accuracy degradation to a minimum – a well known approach is called quantization, consisting on mapping the floating-point values to a fixed-point representation. Moreover, limiting the number of bits to represent the features and model's parameters also reduces the computation resources and consequently leads to energy savings.

Authorized licensed use limited to: b-on: UNIVERSIDADE DO MINHO. Downloaded on April 23,2026 at 15:01:09 UTC from IEEE Xplore. Restrictions apply.

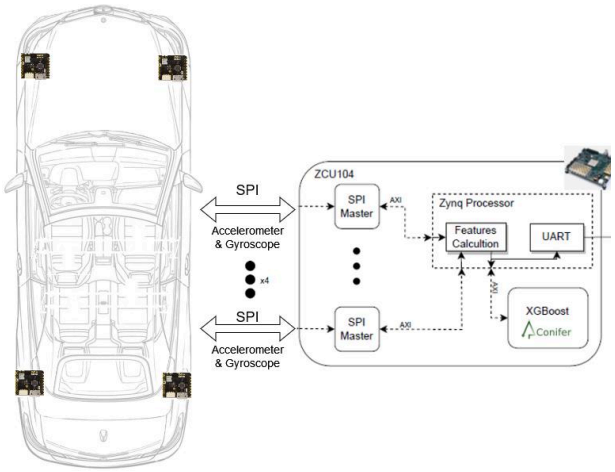


Fig. 5. Connections in the real-time system setup.

Both steps are made internally by Conifer, as the user can choose the total number of bits, and how many should be allocated to the integer and decimal parts. In this context, the following experiments aimed to find the optimal combination of bits that leads to the best balance between precision and performance. The optimal value for the integer portion was achieved by checking the dataset's values distribution, as 8 bits were needed to represent highest absolute value of 154; tests were then made to find the best number of bits for the decimal part, and since the results with 8 and 10 bits were the same, with both being better than the obtained accuracy with the model represented on the default 32-bit precision (87.98%), the lower value of 8 bits was the selected value for the decimal portion. In summary, a total of 16 bits were used, equally distributed for the integer and decimal components, with the model's deployment done on a Xilinx Zynq UltraScale+ MPSoC ZCU104.

IV. Results and Discussion

Fig. 6 reflects the model's confusion matrix on the test set. Despite the difference on the number of samples of both "Pothole Right" and "Speed Bump" classes in comparison to the others, which can lead to a biased interpretation of the overall accuracy, the accuracies per class are defined as:

- No Event: 92.6%.
- Pothole Left: 86.7%.
- Pothole Right: 84.0%.
- Speed Bump: 81.1%.

The performance between the pothole-related classes shows that the model is robust enough to distinguish which side of the car crosses the events. On the other hand, most of the misclassifications happen between an event class and "No Event". From the pothole classes point of view, some events do not cause sufficient vibrations to be detected. On speed bumps, the cause is mostly related with the labelling procedure: a "Speed Bump" event is considered from the moment that the vehicle starts to go up until the moment that it goes out of the speed bump. Nevertheless, the model's generalization capacity is also validated by the conducted approach at the time of splitting the dataset, as all the samples related with a specific location are allocated to a single subset, avoiding any spatial relationship between the different subsets.

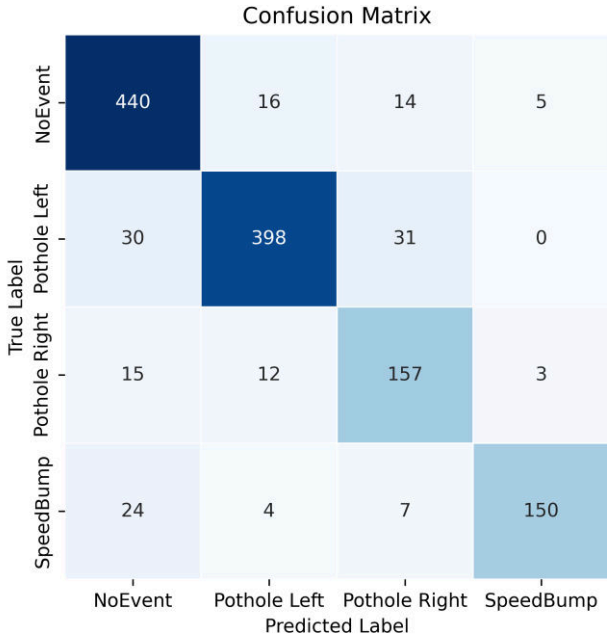


Fig. 6. Confusion matrix of the model converted with Conifer.

The real-time prototype was initially configured to follow the defined configurations at the time of the model's creation (5 second windows with an overlap of 0.5 seconds, with a sensor's sampling rate of 100Hz), as 500 points are acquired to fullfill the input window buffers, with the predictions made on every new 50 points. Over 20 iterations, the system takes 1.38 seconds on average to acquire the new data, 26.15 miliseconds to calculate the required features, and 750 microseconds to perform inference. Given a high window size and a low overlapping period between samples, the initial experiments show that the temporal context of crossing an event extends for too long a period of time – consequently, experiments were made to reduce the number of points that compose a sample, discarding the overlapping step, which is also valid for the model since the 22 input features are calculated as well. These newly conducted tests showed that calculating the features based on 100 points instead of 500 also leads to very good results.

Over the same value of 20 iterations, the feature calculation is optimized to 24.64 miliseconds since the features are calculated on a smaller amount of data, but the acquisition time increases to 2.77 seconds because more points are acquired at each iteration. In terms of resource usage, the model uses 10865 Lookup Tables (LUTs), which corresponds to only 4.72% of all the FPGA's available LUTs. Regarding power consumption, the measured values vary between 10.1 Watts and 10.3 Watts at each iteration.

In summary, all the obtained results are very satisfactory, as the latency is fast enough considering that the use case does not correspond to a critical decision making application that requires an extremely fast response. Moreover, and knowing beforehand that the FPGA's power consumption is around 10W when no processing is being conducted, it leads to the conclusion that the required resources (between 100 and 300 milliwatts) are very low and suitable considering the constraints associated to having such a system on a vehicle.

Authorized licensed use limited to: b-on: UNIVERSIDADE DO MINHO. Downloaded on April 23,2026 at 15:01:09 UTC from IEEE Xplore. Restrictions apply.

V. Conclusion

In this work, we propose a machine learning pipeline that extracts time-based features from 4 IMU sensors mounted near each vehicle's wheel to monitor the existence of road artifacts (potholes and speed bumps) that can cause damage or suspension wear on vehicles.

Considering the context of ADAS systems, where concerns such as power consumption, latency, and memory are inherent, a deep analysis is performed to find the best combination between dataset and model configurations: best sample window length, best sensor sampling rate, optimal model hyperparameters, and best subset of features were subject to different experiments, with the model being then deployed to FPGA through the Conifer framework, with a study related to the optimal balance between performance and precision from which both the input values and the model's thresholds were represented.

The results show that our approach is not only capable of detecting events accurately, but also to distinguish on which side of the vehicle the impacts occur. Despite the very good results in terms of power consumption and inference time with FPGA, there is still potential to optimize both the data acquisition and feature extraction times with the implementation of the features' mathematical operations in HDL, in order to have all the pipeline implemented on the PL side of the FPGA. This approach will mitigate the high latency that comes from the constant communications between PS and PL that is needed on the current pipeline (data acquired from PL is sent to PS, and the calculated features on PS are sent back to PL), which requires an application of a dynamic delay value to ensure that the 100Hz sampling rate is achieved.

Thus, future works should approach a similar workflow on platforms with even lower resources (such as microcontrollers), as well as increase the available data together with a more robust annotation strategy, since it can enhance the model's generalization capabilities.

Acknowledgment

This work was supported by national funds through the Operational Competitiveness and Internationalization Programme (COMPETE 2020) [Project N.º 179491; Funding Reference: SIFN-01-9999-FN-179491], and by FCT (Fundação para a Ciência e Tecnologia) within the Project Scope of the R&D Unit Number 00319.

References

[1] E. Edmonds, "AAA: Potholes Pack a Punch as Drivers Pay $26.5 Billion in Related Vehicle Repairs", 2022, NewsRoom. [Online]. Available: https://newsroom.aaa.com/2022/03/aaa-potholes-pack-a-punch-as-drivers-pay-26-5-billion-in-related-vehicle-repairs/.

[2] N. Haworth, M. Symmons, "The relationship between fuel economy and safety outcomes", 2002, Monash University Accident Research Centre, Report 188. [Online]. Available: https://www.monash.edu/muarc/archive/our-publications/reports/muarc188.

[3] S. Summers et al., "Fast inference of Boosted Decision Trees in FPGAs for particle physics", Journal of Instrumentation, vol. 15, May 2020, doi: 10.1088/1748-0221/15/05/P05026.

[4] A. Shah, G. Sharma, and L. Bhargava, "Smart implementation of computer vision and machine learning for pothole detection", 11h Int. Conf. on Cloud Comp., Data Science & Engineering (Confluence), 2021, doi: 10.1109/Confluence51648.2021.9376886.

[5] B. Nanda Kumar, S. G. Kulkarni, S. Kakkar and K. Vipin, "ADAS Pothole Detection System using FPGA," 2023 IEEE Asia Pacific Conf. On PG. Research In MicroElect. And Elect. (PRIMEAsia), Hyderabad, India, 2023, pp. 56-57, doi: 10.1109/PRIMEAsia60757.2023.00028.

[6] W.-H. Chen, H.-J. Hsu, and Y.-C. Lin, "Implementation of a real-time uneven pavement detection system on fpga platforms" in 2022 IEEE Int. Conf. on Consumer Elect. - Taiwan, 2022, pp. 587- 588, doi: 10.1109/ICCE-Taiwan55306.2022.9869054.

[7] Y.-C. Lin, W.-H. Chen, and C.-H. Kuo, "Implementation of pavement defect detection system on edge computing platform." Applied Sciences, vol. 11, no. 8, 2021, doi: 10.3390/app11083725.

[8] M. H. Asad, S. Khaliq, M. H. Yousaf, M. O. Ullah, and A. Ahmad, "Pothole detection using deep learning: A real-time and ai-on-theedge perspective," Advances in Civil Engineering, vol. 2022, no. 1, 2022, doi: 10. 1155/2022/9221211.

[9] A. Tithi, F. Ali, and S. Azrof, "Speed bump & pothole detection with single shot multibox detector algorithm & speed control for autonomous vehicle," in 2021 Int. Conf. on Autom., Control and Mechat. for Industry 4.0 (ACMI), 2021, pp. 1-5, doi: 10.1109/ACMI53878.2021.9528185.

[10] A. Mednis, G. Strazdins, R. Zviedris, G. Kanonirs, and L. Selavo, "Real time pothole detection using android smartphones with accelerometers." in 2011 Int. Conf. on Dist. Comput. in Sensor Systems and Workshops (DCOSS), 2011, pp. 1-6, doi: 10.1109/DCOSS.2011.5982206.

[11] V. Rishiwal and H. Khan, "Automatic pothole and speed breaker detection using android system," in 2016 39th Int. Conv. on Info. and Comm. Tech., Elect. and Microelect. (MIPRO), 2016, pp. 1270-1273, doi: 10.1109/MIPRO.2016.7522334.

[12] S. Tiwari, R. Bhandari, and B. Raman, "Roadcare: a deep-learning based approach to quantifying road surface quality," in Proc. of the 3rd ACM SIGCAS Conf. on Comp. and Sustainable Societies, 2020, pp. 231-242, doi: 10.1145/3378393.340228.

[13] R. Bustamante-Bello, A. Garca-Barba, L. A. Arce- Saenz, L. A. Curiel-Ramirez, J. Izquierdo-Reyes, and R. A. Ramirez-Mendoza, "Visualizing street pavement anomalies through fog computing v2i networks and machine learning," Sensors, vol. 22, no. 2, 2022, doi: 10.3390/s22020456.

[14] I. Ferjani and S. A. Alsaif, "How to get best predictions for road monitoring using machine learning techniques," PeerJ Computer Science, vol. 8, p. e941, 2022, doi: 10.7717/peerj-cs.941.

[15] O. A. Egaji, G. Evans, M. G. Griffiths, and G. Islas, "Real-time machine learning-based approach for pothole detection," Expert Systems with Applications, vol. 184, 2021, doi: 10.1016/j.eswa.2021.115562.

[16] J. Duarte et al., "Fast inference of deep neural networks in FPGAs for particle physics," JINST, vol. 13, no. 07, 2018, doi: 10.1088/1748-0221/13/07/P07027.

[17] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system" in Proc. of the 22nd ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, Association for Computing Machinery, 2016, doi: 10.1145/2939672.2939785.

[18] K. Druart, D. Espes, C. Dezan, and A. Deturche, "Multiclass random forest on fpga using conifer", 2025, University of Brest. [Online]. Available: https://accml.dcs.gla.ac.uk/papers/2025/7th_AccML_paper_4.pdf.

[19] L. Sousa et al., "Enhancing Automotive Products with TinyML and MEMS Sensors: A Preliminary Approach" in 2025 Data Science and Artificial Intelligence (DSAI 2024). Communications in Computer and Information Science, vol. 2318. doi: 10.1007/978-981-97-9793-6_14.

[20] P. Delgado, C. Melo, G. Carvalho, V. Fukuda and R. Silva, "Road Surface Classification Using an Accelerometer and MiniROCKET on Ultra-Low-Power STM32," 2025 Sixteenth International Conference on Ubiquitous and Future Networks (ICUFN), Lisbon, Portugal, 2025, pp. 234-240. doi: 10.1109/ICUFN65838.2025.11169954.

Authorized licensed use limited to: b-on: UNIVERSIDADE DO MINHO. Downloaded on April 23,2026 at 15:01:09 UTC from IEEE Xplore. Restrictions apply.