Smart Objects: Impact localization powered by TinyML

Ioannis Katsidimas
Computer Engineering and Informatics Department, University of Patras, Greece
ikatsidima@ceid.upatras.gr

Thanasis Kotzakolios
Mechanical Engineering and Aeronautics Department, University of Patras, Greece
kotzakol@upatras.gr

Sotiris Nikoletseas
Computer Engineering and Informatics Department, University of Patras, Greece
nikole@cti.gr

Stefanos H. Panagiotou
Computer Engineering and Informatics Department, University of Patras, Greece
spanagiotou@ceid.upatras.gr

Constantinos Tsakonas
Computer Engineering and Informatics Department, University of Patras, Greece
st1059666@ceid.upatras.gr

ABSTRACT

Growing momentum in embedded systems and the wide use of sensors in everyday life, have motivated significantly, novel research in Internet of Things (IoT) systems and on-device Machine Learning (TinyML) processing. However, limitations in the energy stock and the computational capabilities of resource-scarce devices prevent the implementation of complex ML algorithms in IoT devices, which typically have limited computing power, small memory, and generate large amounts of data. This paper, aims to research and exploit the TinyML emerging technology for embedding intelligence in low-power devices, towards next generation IoT paradigm and smart sensing, in the context of SHM. In particular, the purpose is to provide integrated SHM functionality in plastic structures and thus make them "conscious" and self-explanatory (smart objects), by being able to localize any occurring impacts on the structure. We implement and benchmark Random Forest and Shallow Neural Network models on Arduino NANO 33 BLE, using an experimental dataset of piezoelectric sensor measurements concerning impact events in a thin plastic plate. The classification and model footprint results, 98.71% - 8KB and 95.35% - 12KB of accuracy and flash memory size for each model respectively, are very promising and constitute a solid baseline for motivating our concept.

CCS CONCEPTS

• **Hardware → Sensors and actuators**; • **Computer systems organization → Real-time systems**; • **Computing methodologies → Distributed artificial intelligence**.

KEYWORDS

TinyML, Structural Health Monitoring, Piezoelectric Transducers, Intelligent IoT, Extreme Edge

SenSys '22, November 6–9, 2022, Boston, MA, USA

ACM Reference Format:
Ioannis Katsidimas, Thanasis Kotzakolios, Sotiris Nikoletseas, Stefanos H. Panagiotou, and Constantinos Tsakonas. 2022. Smart Objects: Impact localization powered by TinyML. In *Workshop on Challenges in Artificial Intelligence and Machine Learning for Internet of Things (AIChallengeIoT) (SenSys '22), November 6–9, 2022, Boston, MA, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3560905.3568298

1 INTRODUCTION

Recent advances in algorithms, hardware and software have facilitated the realization of performing on-device sensor data analytics, namely TinyML[8] (also known as extreme edge ML or embedded ML), in low power and limited resources IoT devices. Mainly focusing, but not limited to applications that are related to; i) computer vision ii) audio and speech processing, iii) natural language processing, and iv) activity recognition, has set a strong basis for a lot more potential modern systems that promote principles such as decentralization and cost effectiveness. TinyML can be also characterised as the epitome of utilisation and resources availability, as it successfully demonstrates the most effective use of models and resources, to achieve the acceptable performance to the service provided. TinyML has also contributed to the realization of novel paradigms such as smart objects and smart sensing, presenting a new era in the Internet of Things (IoT) research area, where embedded devices present advanced intelligence capabilities, resulting to Artificial Intelligence of Things (AIoT) or Intelligent IoT (IIoT). Structural Health Monitoring (SHM) plays a vital role in decision making concerning maintenance operations and strategies, making it a very wide application subject with many challenges to apply sensing and processing capabilities. Core element to a modern digitalized SHM system is sensing, where in most cases include vibroacoustic, piezoelectric, stain gauges, fiber optics, cameras (for image analysis), fluid (lubricant, oil) pressure and consumption, accelerometers, and thermography sensors. Vibration-like sensors are the most widely used in SHM systems as they enable vibration detection that results to an equivalent electrical signal that feeds a processing unit. Vibration analysis is an accepted and reliable method for monitoring the operation and performance of the structures, while its transparent adoption without any process interference offers a sustainable monitoring solution. Most common signal processing methods include Fourier transformation (and its variants such as fast and short-time), statistical analysis

over time-series, Cohen's class, wavelet transform, Hilbert-Huang transformation, Neural Networks (NN), Bayesian classifiers and further hybrid approaches [10]. Finally, there are typically two approaches that mainly attempt to deal with SHM problems [13]; i) knowledge-based or model-driven models, which in some cases either neglect realistic and real life conditions or make some convenient assumptions, and ii) data-based models, which suffer from sufficient data availability, as running multiple times an experiment is costly in time, equipment and personnel.

2 RELATED WORK

To the best of our knowledge, there is a gap in the literature concerning impact localization methodologies that use TinyML techniques and are deployed in resource constrained edge devices. Therefore, in this section we present some research works which can be considered as the most related to the current paper. The section is divided in two parts, covering related work in plate structure impact localization based on machine learning techniques and data from piezoelectric sensors (PZTs) and research related to TinyML applications in the industrial context.

2.1 Impact Localization

Dipietrangelo et al., [7] leveraged Polynomial Regression and a Shallow Neural Network (SNN) to detect low speed impacts on an aluminium plate. To perform the experiments, the authors acquired the impact signals using four PZT sensors connected to oscilloscope and dropping a steel ball. The algorithms were compared using the Mean Radial Error metric (MRE), resulting in a MRE of 1.5mm and 1.2mm for the two models respectively. In addition, Balasubramanian et al., [2] compared the accuracy of Artificial Neural Network (ANN), Convolutional Neural Network (CNN), and Long Short-Term Memory (LSTM) network in estimating impact location from PZT sensor response in a aluminum plate. The networks were compared with Mean Absolute Error (MAE) and the results indicate that the ANN gives better accuracy with an MAE of 22 mm, in comparison to CNN (MAE = 31 mm) and LSTM (MAE = 25 mm). Similarly, Datta et al., [5] demonstrate a least square support vector regression-based (LS-SVR) algorithm to localize impact event in terms of X- and Y-coordinates (and its energy) on a CFRP plate-like structure. The mean error for X coordinate was 18.36 mm and for Y coordinate 21.88 mm, performing better than other algorithms reported in literature. Hesser et al, [11] propose a machine learning approach to identify the presence of active sources and to determine the source coordinates of ball impact events, in an aluminum plate equipped with PZT sensors. A finite element model is developed in order to simulate numerical data for the impact events and train Artificial Neural Network (ANN) and Support Vector Machine (SVM) algorithms. Then, experimental data were used only to test and evaluate the performance of the trained models, from which the ANN was considered the best model resulting in average error of 2.7 mm. Tabian et al., [17] report on a novel metamodel for impact detection, localization and characterization of complex composite structures based on Convolutional Neural Networks (CNN) and passive sensing. An innovative methodology was proposed for transforming the raw data acquired from a network of PZT sensors, to 2D images which can be given as input into a CNN and the model accuracy reached values between 94.3% and 100% .

2.2 TinyML in Industrial Applications

Zonzini et al., [18] explored the application of TinyML for Vibration-Based Structural Health Monitoring scenarios. In particular, the authors explored the deployment of the One Class Classifier Neural Network (OCCNN), into a resource-constrained device (Arduino Nano 33 BLE Sense). The OCCNN was ported on the device and validated with experimental data from the Z24 bridge use case, reaching an average accuracy and precision of 95% and 94%, respectively. Athanasakis et al., [1] studied the application of TinyML techniques for optimizing machine learning models in the context of remaining useful life (RUL) prognosis. The authors used the C-MAPSS NASA dataset composed of simulation time series sensor data, STMF767ZI microcontroller and the X-CUBE-AI tool, to predict the RUL of turbofan engines using several models, namely LSTM, CNN, XGBoost and Random Forest. The results demonstrate that it is possible to deploy quantized CNN models with as low as 26KB of Flash and 9KB of RAM memory, with an average 10% quality loss in the RMSE metric for embedded devices. Pau et al., [15] compared different artificial neural networks (ANNs) topologies for predicting the state of charge of batteries, using the IEEE DataPort dataset that contains measurements and features from 72 real driving trips recorded with the electric BMW i3 car. The ANN models (namely Dense NN, CNN+Dense, LSTM, GRU) were tested and validated, demonstrating their effectiveness, on automotive microcontroller SPC584B by using the toolset SPC5-Studio.AI. Bratu et al, [15] propose a low-powered unsupervised learning solution that can detect anomalies in the vibration patterns of bearings. It uses an Autoencoder that takes as input the median absolute deviation of each measurement set provided by an accelerometer, and then uses a classifier that compares the values provided by the output to values that are known to be normal vibration patterns, and then makes a decision for the occurrence of an anomaly. The Autoencoder is fitted on an ESP32 board with TensorFlow Lite, predicting unseen data with an accuracy of 93.42%.

Contributions of this work. Differentiating from the aforementioned research works, in this paper we intend to complement the literature with the following contributions, which we are the first, to the best our knowledge, to publish:

- On-device intelligence (powered by TinyML) to solve impact localization problem of a thin plate, by introducing two models that perform effectively w.r.t. TinyML principles.
- Real-time sensor data collection with MCU (as opposed to the usage of Oscilloscopes), that achieves and demonstrates, high sampling frequency (100 KHz) and extreme low latency (8.5 μs) while using multiple channels of the ADC with Arduino NANO.
- First exploitation of the publicly available Impact Events dataset [12].

3 METHODOLOGY

3.1 Problem Definition

This paper introduces a real-time solution, which will accomplish the impact localization at the edge utilizing TinyML models. Thus,

we consider the following problem definition:
Impact Localisation Multiclass Classification Problem. Given a plate P and a set of piezoelectric sensors S, determine the area Q that an impact event occurs.

Similar studies address this problem as a regression task, yet the proposed solutions are not designed or demonstrated for real-time monitoring of the structure w.r.t. IoT constrains. Thus, in our work the problem is treated as a classification task, in order to prove the concept of real-time smart monitoring using a low-cost system.

3.2 Setup and Design of Experiments

3.2.1 Structure Model. The structure that is used is a thin acrylic plate P with dimensions of 300mm x 300mm and 4mm thickness, which is separated in 25 identical 60mm x 60mm square tiles, onto which four Piezoelectric transducers (PZTs) are bonded at the corners. In particular, the set of the square tiles is denoted as $Q = \{q_{1,1}, q_{1,2}, ..., q_{1,5}, q_{2,1}, q_{2,2}, ..., q_{2,5}, q_{3,1}, ..., q_{5,4}, q_{5,5}\}$ where x and y at $q_{x,y}$ is the row and column tile respectively, thus PZT A is deployed at $q_{1,1}$, PZT B at $q_{1,5}$, PZT C at $q_{5,5}$, and PZT D at $q_{5,1}$ (and they are denoted as sensor[A|B|C|D] in the following sections). The impact event is produced by a steel ball B (9.5 mm diameter, 3.53 gr weight), which is released from a height distance that varies from 10 cm to 20 cm with 0.5 cm interval, from the P, and performs a free-fall motion.

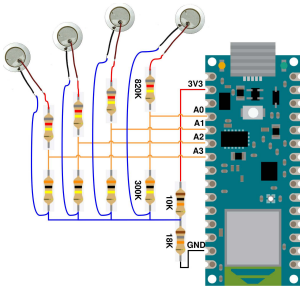

Figure 1: PZT modules connected to Arduino NANO 33 BLE.

3.2.2 Monitoring System. In order to monitor the structure, acquire the propagated wave, and make the impact localization, the piezoelectric sensors are bonded to the structure and connected to a microcontroller (Figures 1 and 2 depict the sensing setup) that is responsible to perform impact localization services. The controller is an Arduino NANO 33 BLE, and is selected due to the memory capacity and processor speed it provides. It is also one of the suggested devices for TinyML applications, while being acceptable both from the industry and the research community. Moreover, it is compatible with the most state-of-the-art TinyML frameworks. The voltage divider between the sensors and the device shifts the reference voltage of the idle state to a value greater than zero. This further enables the data acquisition, since as soon as the impact occurs, the structure flexes, producing positive and negative output voltage by the PZTs. On the microcontroller's side, the hardware does not support negative values reading, and therefore, the voltage divider fixes this issue.

4 DATA PROCESSING APPROACH

4.1 Dataset Outline

The data [1] used in this paper comes from the Impact Events dataset [12]. The conducted experiments are the least possible in order to cover the sample space and assure good convergence for the machine learning algorithm. The points of impact for the data collection are provided by the Sobol algorithm [3]. Each experiment is denoted by its (x, y, h) coordinates and is repeated at least three times in order to enhance the dataset, but also ensure the robustness of the experiment. The number of acquired samples for each experiment is 5000 samples for each sensor. The final dataset includes 771 instances (multiple repetitions for 200 distinct impacts) and the labels are the corresponding squares $q_{x,y}$ based on the experiment's (x, y) coordinates. An overview of the dataset's format is depicted at Table 1.

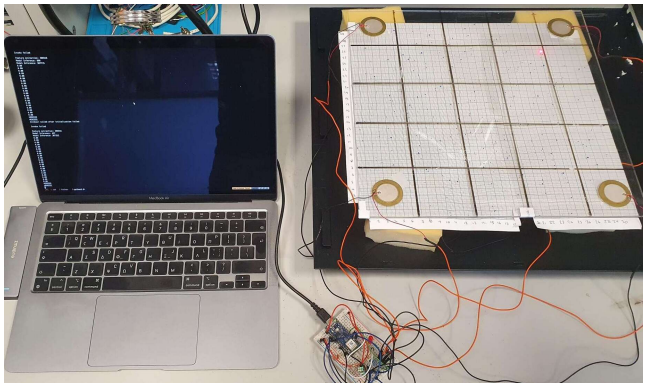

Figure 2: Sensing setup overview. The laptop is used to display the console for the arduino's output. The impact identification model runs on the arduino device.

ExperimentNo	SensorA	SensorB	SensorC	SensorD	x	y	height	position

Table 1: Dataset format

4.2 Low-Latency Sensing Solution

As mentioned before, the data is collected using the microcontroller, which sends over the UART bus the sensed values from the PZTs to store them. The sensors are connected to the analog inputs of the device and the sampling frequency of the Analog-to-Digital Converter (ADC) is set at 100KHz, to ensure that the impact phenomenon is captured as detailed as possible. The Arduino can achieve up to 200KHz, however, when sampling predefined number of samples with that frequency, important information at the end of the impact is not captured.

For the signal recording, an auto-trigger mechanism is developed on the device to initiate the acquisition procedure. The primary purpose of the trigger mechanism is to reduce the power consumption and inference time of the device. This has been achieved by reducing the memory transactions and the processor's operations during the impact anticipation stage.

The whole process of the sensors reading is implemented on the hardware level, exploiting the Arduino's Successive Approximation ADC (SAADC), and thus the processor is not responsible for enabling the analog channels and storing the values in the memory repeatedly. Although SAADC's sampling rate is over the mark of 15KHz and operates with multi-channel mode, it is important to note that the Arduino introduces memory writing abnormalities, due to synchronization problems updating SAADC's storing buffer

[1] Also available at https://github.com/Smart-Objects/Impact-Events-Dataset

index and size. Hence, the main solution to overcome this problem is to transfer the data to a second data structure as soon as reading of all the available channels is finished and then restart the SAADC to reinitialize the buffer. However, we exploit the Programmable Peripheral Interface (PPI), interconnecting the SAADC and a Timer to synchronize the needed operations to update the size of the SAADC's buffer using timer triggers. By using the PPI, the SAADC's restarting is avoided.

Everything regarding the sampling procedure is implemented on the hardware level for efficiency. The only action that the processor performs at this stage is to check if the relative change in two successive readings is greater than a threshold set to 10%. This trigger aspect enables the inferences to run intermittently due to energy limitations, while monitoring the structure. Overall, the sampling latency (setting and initializing the ADC plus storing the data) is 8.5 μs.

4.3 Feature extraction and selection

As already mentioned, our aim is to create a computational lightweight data processing and ML methodology for resource constrained devices. Concerning the raw data filtering, we do not apply FFT, Butterworth filtering or any other preprocessing techniques proposed in the literature as these should also be applied in the Arduino MCU before operating the ML inference, and thus they would result in a significantly higher computational complexity. At the same time, considering the highly accurate results in Section 6, the benefits in accuracy that would occur by using such techniques, in the present work, would be low and disproportionate to the cost of the higher complexity. However, we argue that this may stand as a future point of research.

Regarding feature extraction, we experiment with several statistical measures, given that they are computationally suitable for the MCU, from the Impact Events dataset. The dataset contains 128 features in total (32 features for each of the 4 sensors) and our aim is to select an optimal number of features from this set, to speed up the on-device data processing pipeline as well as the ML model size and inference.

To select the features, we perform the following steps. First, we calculate the Pearson correlation amongst all the features and eliminate those whose correlation exceeds the threshold of 0.9, in order to keep only uncorrelated features in the dataset. Also, we remove all the zero-variance, i.e., features that have the same value in all samples. Afterwards, we calculate the ANOVA[2] F-values for the remaining features and eventually we select, examining also their computational complexity, the following feature groups (statistical measures), for each sensor: Absolute Maximum Value, Minimum, Standard Deviation, Kurtosis and Skewness.

Eventually, our processed dataset consists of the target feature describing the impact area and thus contains 21 classes/areas, as well as 20 input features in total (5 statistical measures x 4 sensors), which are also implemented in C code in the MCU and calculated for each impact experiment.

[2]ANOVA is well-performed and accepted method for feature importance ranking

5 MODELS AND ALGORITHMS

The proposed solution runs on the extreme edge and in particular on an IoT device, providing real-time information for impact events on the structure, by specifying the area of interest they occurred. Thus, the selected model's size and inference latency should be as efficient as possible, in order to achieve real-time monitoring, and accurate predictions on the output.

The models that have been tested, considering the inference time and memory footprint, are a Random Forest Classifier (RF), XGBoost, Support Vector Machines (SVM) and a Shallow Neural Network. We note that our modeling approach is structure-agnostic, meaning that the impact localization problem is solved in a pure data-driven fashion and independently of the material and geometry of the structure, as the models are trained with the sensor data and do not leverage any physics-based attributes. To train and evaluate the models, the data is splitted (by shuffling and stratified fashion) into training and test sets with a 70/30% ratio.

5.1 Random Forest

Random Forest (RF) is an ensemble of two or more decision trees, called estimators, which focuses to eliminate the disadvantages a decision tree introduces [4]. Such disadvantages are over-fitting and low predictive accuracy. Each tree, which constructs the RF, is obtained by randomly selecting a subset from the dataset and create the decision tree based on it. Samples of the dataset can be included in more than one subset. Constructing the decision trees with that procedure ensures that the RF consists of uncorrelated trees, reducing the risk of over-fitting, and invariability to outliers and noise. This method is known as bootstrapping. The output the RF produces is deducted by Bootstrap Aggregation (Bagging). For a classification task, bagging is, essentially, the majority voting of each particular estimator.

The general structure of the RF model is defined by three main hyperparameters, depth of each tree, numbers of estimators and number of sampled data. In general RF's complexity scales at a large rate when the estimators and depth of each tree increases. However, reducing the initial features by feature engineering, translates to a huge reduction to the needed estimators and depth. Moreover, one major advantage of Random Forests over Neural Networks is the explainability aspect, which provides crucial information about the factors that determine each prediction. This is especially important for critical domains such as SHM applications that determine the health status and longevity of the structures.

After experimenting with the hyperparameters of the RF model, the combination that yields an accuracy over 95% comes from the following setting: i) Estimators: 30, ii) Depth: 8, and iii) Sampled Data: 280.

5.2 Shallow Neural Network

Overall, the model architecture should have an optimal balance concerning computational complexity and memory occupation, while keeping an acceptable accuracy ratio. In general, there are two main approaches to achieve this [14]. The first is to create a small and sufficient architecture and gradually augment it until the accuracy criteria are met. On the other hand, the second approach follows the opposite route, i.e, to create a complex and highly accurate DNN

or CNN and then use TinyML techniques to optimize the model so it fits on the microcontroller alongside the rest of the operations. As a first step, in the current study we follow the first approach, and implement a Shallow Neural Network (SNN).

Shallow NNs are NNs that consist of one or two hidden layers, although combining them with the selected features can provide a trained model with a low number of parameters and high accuracy. Since, the inputs are the extracted features, which means their scales are different, standardization needs to be applied on the data, otherwise the features are not comparable to one another. If standardization is not applied there are chances of higher weightage to features with higher magnitude. Given that the number of hidden layers is decreased, it ensures that inference time and memory footprint are not prohibitive.

The best architecture that provided the highest accuracy is depicted at Table 2. The final model consists of 4,117 parameters with float 32 precision. The one and only optimization method we applied is quantization with float 16 (the model is denoted later on as Q-SNN), in order to convert the model to an MCU compatible and exploitable form. Quantization is the procedure to reduce the numerical precision, when storing tensors or executing operations. This yields compact models and lower inference time, whereas the model's accuracy reduction is insignificant and this is due to the low number of parameters at the initial architecture. Also, one of the main techniques for model reduction is pruning, as it removes weights from neurons, and this results to less parameters and model size. However, the application of pruning methods is avoided in our case as we notice in our experiments a rapid drop in the model's accuracy.

Layer	Units	Activation
Input	20	-
Dense	64	relu
Dense	32	relu
Dense	21	softmax

Table 2: Shallow Neural Net Architecture

5.3 Gradient Boosted Trees and Support Vector Machines

Gradient Boosted Trees model is an ensemble of decision trees, like Random Forest. The individual trees are created in a sequential manner and the idea behind these trees is to combine individual trees with high loss (weak learners), in order to create a decision tree that minimizes the loss function (strong learner). The goal of each tree is to minimize the error of the previous tree. At each iteration of the algorithm, a decision tree is appended, until there are no further improvements. Gradient Boosted Trees (GBTs) is one of the most capable machine learning algorithms, however, it is prone to over-fitting, and the memory footprint and the inference time is high, due to the size of the ensemble and the hyperparameters that need to be stored. In this paper, we experiment with XGBoost, which is one of the fastest implementations of GBTs.

Support Vector Machines (SVM), statistical learning model that aims to provide the best margins between the discrete classes, in order to classify the data, whereas when used for regression it provides the best margins that encapsulate the data. When used for high dimensional data, SVMs map the data to hyperplanes to provide the hyperplane with the largest separation of the data points. To achieve the data transformation, SVM uses kernel functions. SVM is very sensitive to noise and is difficult to interpret the final model. A first assumption is that SVM and XGBoost models will not be optimal solutions in our context due to their aforementioned disadvantages. However, we opt to benchmark them as they are one of the most commonly used and effective algorithms for tabular data [16] and thus they provide a comparison basis for our case.

5.4 On-Device Model Deployment

To realize the on-device ML model operation, we first train the models in our personal workstation and then deploy them in the Arduino to perform the inference. Concerning the actual model conversion in an appropriate format for the MCU, there is not any single library that can be used for both the traditional ML algorithms and the Neural Networks.

In specific, to convert the former models (RF, XGBoost, SVM), we use the micromlgen library [9]. This library converts the first two models into C++ header file using multiple *IF-ELSE* statements to represent the decision graph and for the SVM conversion it also produces a C++ header file that includes the needed kernel operations. Concerning the SNN, the TensorFlow Lite (TFLite) library [6] is utilized, which introduces quantization, pruning and several other optimizations methods. TFLite converts the initial model to an optimized FlatBuffer format, and the optimizations occur on the converted model. After the model is in TFLite form, a C++ header file is generated with the hex representation of the model by using xxd command on Unix. This header file is later manipulated by the TensorFlow Lite Micro C++ API on the microcontroller to invoke the inference.

Finally, to calculate the footprint metrics, the model size for RF, XGBoost, SVM and SNN corresponds to the size of the joblib file, since no optimizations occur or size reduction, and .tflite file respectively. Also, for the inference (prediction) time of each model we average the inference timings over the test set instances.

6 EXPERIMENTAL EVALUATION

To evaluate the models we mainly focus on footprint metrics such as inference time and memory size, and quality metrics such as classification accuracy, F1-score, precision and recall.

The results are summarized in Table 3. Overall, we conclude that RF is the best model to be selected, as it is clearly better from the SNN and Q-SNN across all the footprint and quality metrics. It is true that the NN models are close to RF, however the latter also offers explainable insights by design, a crucial criterion for applications such as the one studied in this work. Therefore, in our case the RF is the preferred model, considering the application requirements, problem definition and data characteristics.

We note that the high accuracy scores are achieved by using only 540 instances (experimental impacts) for training the models (the rest 231 instances belong to the test set). This proves that a careful design of experiments, like the presented one which predefines the impact experiments using the Sobol algorithm, can lead to qualitative and representative data, and thus in high ML model accuracy as well as indirectly in less model size.

Concerning the model footprint, despite the timing differences the inference time of all models is acceptable for the real-time

Model	Inference Time	Inference Time+Feature Extraction	Model Size	Accuracy	F1-score	Precision	Recall
Random Forest	0.000284s	0.323s	8KB	98.71%	0.99	0.99	0.99
SVM	0.06825s	0.377s	168KB	90.95%	0.91	0.93	0.91
XGBoost	-	-	835KB	93.97%	0.94	0.95	0.93
Shallow NN (SNN) with float 32	0.000911s	0.379s	20KB	96.98%	0.97	0.98	0.97
Q-SNN: Quantized SNN with float16	0.000265s	0.318s	12KB	95.35%	0.95	0.96	0.95

Table 3: Quality and footprint evaluation metrics for all models

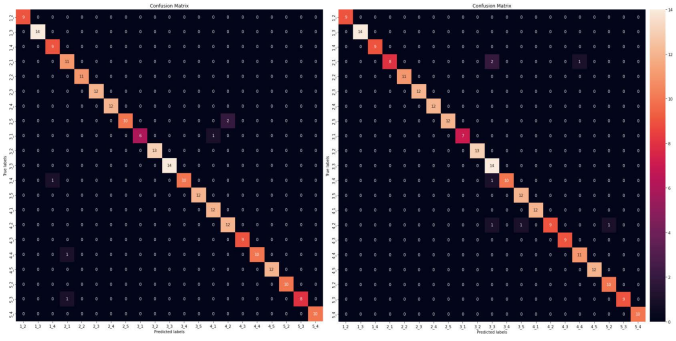

Figure 3: RF's (left) and SNN's (right) Confusion Matrices

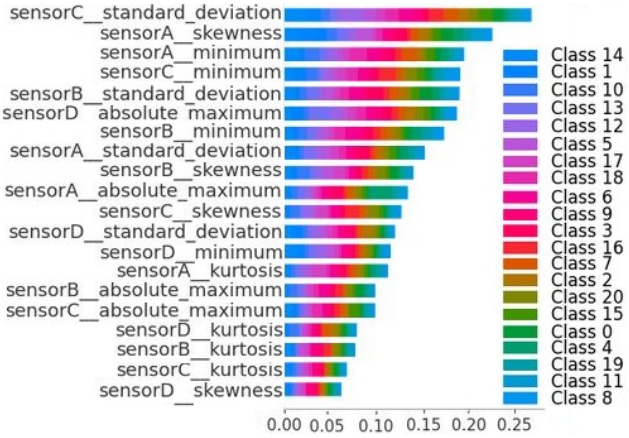

Figure 4: RF's Feature Importance with Shapley Values

requirements of the studied task Additionally, the memory demand of RF and Q-SNN models is very low and insignificant considering the specifications of our MCU. In fact, the present models can be also fitted in even more resource constrained MCUs. However, the XGBoost model has a huge memory footprint, that makes it unfeasible to deploy on the MCU. Similarly, the SVM model is very large in this context but it can be deployed on the MCU, although it is not optimal w.r.t. MCU's available memory for other data storing operations. Concerning the optimization of the Shallow NN with the quantization method, we get remarkable results, a 40% decrease in model size, with only 2% drop in accuracy.

Furthermore, at Figure 3, the classification results in the test dataset are presented, while Figure 4 depicts the feature importance of the RF using SHAP values. In the latter, we notice that the most important features are slightly scattered, i.e., they are not grouped and ordered by each single statistical measure (e.g. standard deviation). This means that the combinations of each sensor and statistical measure have different weights in the classification and they are not uniform across all sensors. However, the most uniform statistical measures are the minimum and standard deviation.

According to the results and our experience, using feature engineering and traditional ML practices is a useful strategy towards real-time TinyML-based IoT systems, even though it is (justifiably, for the known reasons) not the most popular approach in the literature, especially for TinyML research. Our rationale, is to examine all possible solutions and keep the simplest and most effective ones. We suggest following the aforementioned approach depending on the application and data requirements, complimentary with ANN modeling and exploration. Indeed, in the future we intend to experiment more with ANNs (using raw signal data without features as well), given their inherent potential to reach optimal architectures with the highest accuracy and smallest footprint, by leveraging optimizations methods such as pruning, quantization, knowledge distillation etc., whereas conventional ML models have low potential to be optimized when they are already large in size.

To complete the benchmarking w.r.t. to multiclass classification and TinyML optimizations, we also increase the classes by discretizing the structure's square tiles even more. In specific, from 21 classes we result to 56 and 126 classes, giving 95.23% and 94.53% accuracy for the SNN respectively. However, when quantizing the model to 16-bit float the accuracy drops drastically. This is a useful insight about the importance of the 32-bit weight representation, in the case of training with a small dataset and an increase in the number of classes.

7 CONCLUSION

In this paper, we present a low-cost, resource-frugal IoT solution powered by TinyML technology to solve impact localization problem of a thin plastic plate, in real-time conditions. In particular, we developed and configured the IoT hardware, we conducted the feature engineering and data processing, and finally deliver two models that enable on-device impact localization services. The results concerning the classification accuracy and model size, are up to 98.71% - 8KB for the RF and 95.35% - 12KB for the SNN and thus motivate the concept of our implementation for real-time, embedded intelligence in SHM applications and indicate promising potential for further exploration in SHM-related problems and associated TinyML research. We also note that our methodology's properties, including the topics of sensor data collection and processing, allow to derive challenges and relate the findings to a variety of other use cases in research and industry. Therefore, we argue that our approach and system implementation to address the problem studied, is transferable or at least can serve as a reference to other industrial IoT and TinyML applications as well.

ACKNOWLEDGMENTS

The present work was financially supported by the Foundation "Andreas Mentzelopoulos".

REFERENCES

[1] Georgios Athanasakis, Gabriel Filios, Ioannis Katsidimas, Sotiris Nikoletseas, and Stefanos H. Panagiotou. 2022. TinyML-based approach for Remaining Useful Life Prediction of Turbofan Engines. *27th International Conference on Emerging Technologies and Factory Automation (ETFA)* (2022).

[2] Prabakaran Balasubramanian, Vikram Kaushik, Sumaya Y Altamimi, Marco Amabili, and Mohamed Alteneiji. 0. Comparison of neural networks based on accuracy and robustness in identifying impact location for structural health monitoring applications. *Structural Health Monitoring* 0, 0 (0), 14759217221098569. https://doi.org/10.1177/14759217221098569 arXiv:https://doi.org/10.1177/14759217221098569

[3] Paul Bratley and Bennett L. Fox. 1988. Algorithm 659: Implementing Sobol's Quasirandom Sequence Generator. *ACM Trans. Math. Softw.* 14, 1 (mar 1988), 88–100. https://doi.org/10.1145/42288.214372

[4] Leo Breiman. 2001. Random forests. *Machine learning* 45, 1 (2001), 5–32.

[5] Amitabha Datta, M. J. Augustin, Nitesh Gupta, S. R. Viswamurthy, Kotresh M. Gaddikeri, and Ramesh Sundaram. 2019. Impact Localization and Severity Estimation on Composite Structure Using Fiber Bragg Grating Sensors by Least Square Support Vector Regression. *IEEE Sensors Journal* 19, 12 (2019), 4463–4470. https://doi.org/10.1109/JSEN.2019.2901453

[6] Robert David, Jared Duke, Advait Jain, Vijay Janapa Reddi, Nat Jeffries, Jian Li, Nick Kreeger, Ian Nappier, Meghna Natraj, Tiezhen Wang, Pete Warden, and Rocky Rhodes. 2021. TensorFlow Lite Micro: Embedded Machine Learning for TinyML Systems. In *Proceedings of Machine Learning and Systems*, A. Smola, A. Dimakis, and I. Stoica (Eds.), Vol. 3. 800–811. https://proceedings.mlsys.org/paper/2021/file/d2ddea18f00665ce8623e36bd4e3c7c5-Paper.pdf

[7] F. Dipietrangelo, F. Nicassio, and G. Scarselli. 2023. Structural Health Monitoring for impact localisation via machine learning. *Mechanical Systems and Signal Processing* 183 (2023), 109621. https://doi.org/10.1016/j.ymssp.2022.109621

[8] Dr. Lachit Dutta and Swapna Bharali. 2021. TinyML Meets IoT: A Comprehensive Survey. *Internet of Things* 16 (2021), 100461. https://doi.org/10.1016/j.iot.2021.100461

[9] eloquentarduino. 2020. micromlgen library repository. https://github.com/eloquentarduino/micromlgen

[10] Deepam Goyal and Bs Pabla. 2015. The Vibration Monitoring Methods and Signal Processing Techniques for Structural Health Monitoring: A Review. *Archives of Computational Methods in Engineering* (03 2015). https://doi.org/10.1007/s11831-015-9145-0

[11] Daniel Frank Hesser, Georg Karl Kocur, and Bernd Markert. 2020. Active source localization in wave guides based on machine learning. *Ultrasonics* 106 (2020), 106144. https://doi.org/10.1016/j.ultras.2020.106144

[12] Ioannis Katsidimas, Thanasis Kotzakolios, Sotiris Nikoletseas, Stefanos H. Panagiotou, Konstantinos Timpilis, and Constantinos Tsakonas. 2022. Dataset: Impact events for Structural Health Monitoring of a thin plate. In *Proceedings of the 20th ACM Conference on Embedded Networked Sensor Systems*. https://doi.org/10.1145/3560905.3567764

[13] Arman Malekloo, Ekin Ozer, Mohammad AlHamaydeh, and Mark Girolami. 2021. Machine learning and structural health monitoring overview with emerging technology and high-dimensional data source highlights. *Structural Health Monitoring* (2021), 14759217211036880.

[14] Gaurav Menghani. 2021. Efficient Deep Learning: A Survey on Making Deep Learning Models Smaller, Faster, and Better. https://doi.org/10.48550/ARXIV.2106.08962

[15] Danilo Pau, Davide Denaro, Giambattista Gruosso, and Achraf Sahnoun. 2021. Microcontroller architectures for battery state of charge prediction with tiny neural networks. In *2021 IEEE 11th International Conference on Consumer Electronics (ICCE-Berlin)*. 1–6. https://doi.org/10.1109/ICCE-Berlin53567.2021.9720020

[16] Ravid Shwartz-Ziv and Amitai Armon. 2022. Tabular data: Deep learning is not all you need. *Information Fusion* 81 (2022), 84–90. https://doi.org/10.1016/j.inffus.2021.11.011

[17] Iuliana Tabian, Hailing Fu, and Zahra Sharif Khodaei. 2019. A Convolutional Neural Network for Impact Detection and Characterization of Complex Composite Structures. *Sensors* 19, 22 (2019). https://doi.org/10.3390/s19224933

[18] Federica Zonzini, Francesca Romano, Antonio Carbone, Matteo Zauli, and Luca De Marchi. 2021. Enhancing vibration-based structural health monitoring via edge computing: A tiny machine learning perspective. In *Quantitative Nondestructive Evaluation*, Vol. 85529. American Society of Mechanical Engineers, V001T07A004.