Machine-Learning-Based Microwave Sensing: A Case Study for the Food Industry

Marco Ricci, *Member, IEEE*, Bernardita Štitić, Luca Urbinati, *Graduate Student Member, IEEE*, Giuseppe Di Guglielmo, *Member, IEEE*, Jorge A. Tobón Vasquez, *Member, IEEE*, Luca P. Carloni, *Fellow, IEEE*, Francesca Vipiana, *Senior Member, IEEE*, and Mario R. Casu, *Senior Member, IEEE*

***Abstract*—Despite the meticulous attention of food industries to prevent hazards in packaged goods, some contaminants may still elude the controls. Indeed, standard methods, like X-rays, metal detectors and near-infrared imaging, cannot detect low-density materials. Microwave sensing is an alternative method that, combined with machine learning classifiers, can tackle these deficiencies. In this paper we present a design methodology applied to a case study in the food sector. Specifically, we offer a complete flow from microwave dataset acquisition to deployment of the classifiers on real-time hardware and we show the effectiveness of this method in terms of detection accuracy. In the case study, we apply the machine-learning based microwave sensing approach to the case of food jars flowing at high speed on a conveyor belt. First, we collected a dataset from hazelnut-cocoa spread jars which were uncontaminated or contaminated with various intrusions, including low-density plastics. Then, we performed a design space exploration to choose the best MLPs as binary classifiers, which resulted to be exceptionally accurate. Finally, we selected the two most light-weight models for implementation on both an ARM-based CPU and an FPGA SoC, to cover a wide range of possible latency requirements, from loose to strict, to detect contaminants in real-time. The proposed design flow facilitates the design of the FPGA accelerator that might be required to meet the timing requirements by using a high-level approach, which might be suited for the microwave domain experts without specific digital hardware skills.**

***Index Terms*—Microwave sensing, microwave antenna arrays, machine learning, MLP classifier, food safety, high level synthesis, food technology, contactless diagnostics**

I. Introduction

The quest for the highest quality in all the production phases is the key to the success of many industries. The packaging phase is critical for two reasons. First, it can be a source of contamination; secondly, detecting the contaminants after the packaging phase requires methods that inspect inside the package without contact, return a classification (positive/negative) with the highest confidence, and work in real-time at the speed of fast production lines.

The food industry is a paradigmatic example. Due to the continuous growth in the number of mechanized processes for food preparation, the occurrence of physical contamination is increasing [1]. The sources can be multiple, from equipment or packaging failures, to food handlers' inattention, which cause contamination from a variety of different materials. The consequences of not detecting *all* of these contaminants range from customer discontent, which can lead to economic consequences for brands, to severe injuries, like dental issues or choking, dangerous in particular for children and seniors, or even to contaminated intrusions that may contain bacteria.

As a matter of fact, the most widespread method to detect foreign objects in packaged food, X-rays technology [2], is not perfect, even in its latest dual-energy incarnation [3]; its weak spot are the contaminants made of low-density materials, such as polyethylene or polypropylene, which are plastic materials particularly used for packaging. Furthermore, X-rays inspection can be harmful for operators due to the ionizing radiations. Other inspection methods currently used in the food industry, like metal detectors (MD) or near-infrared (NIR) imaging, have other limitations: MD can only detect metallic objects, while NIR have a low penetration depth and a high absorption in water.

The food industry is interested in the development of novel approaches to address these deficiencies. Electromagnetic (EM) sensing represents an interesting option, which uses the information contained in waves reflected by or transmitted through a target, to assess the presence of defects. The method is applicable in different regions of the EM spectrum. Recently, applications using Terahertz (THz) radiations have been developed and applied in industries [4]–[6]. However, the analysis of reflected THz waves can be useful to detect superficial faults only, due to their intrinsic limited penetration depth. On the contrary, micro-waves (MW) in the Gigahertz (GHz) region can penetrate an object up to a few centimeters, making MW-based devices suitable for monitoring products contained in non-metallic packages. Most importantly, MW can detect low-density plastic contaminants.

MW sensing exploits the difference of dielectric properties of two materials in contact, which is commonly referred to as *dielectric contrast*. This difference creates a discontinuity that scatters an incident wave more or less depending on the amount of dielectric contrast [7], [8]. By illuminating a target with low-power microwaves emitted and then captured by multiple antennas, the analysis of the captured signals can detect anomalies present in the target.

Fig 1(a) depicts a scenario in which a MW Sensing Equipment for signal generation and acquisition is positioned along a packaging line. The MW equipment is connected to a

Manuscript received XXX, 2021; accepted XXX. Date of publication XXX; date of current version XXX.

M. Ricci, L. Urbinati, J. A. Tobon Vasquez, F. Vipiana, and M. R. Casu are with the Department of Electronics and Telecommunications, Politecnico di Torino, 10129 Torino, Italy (e-mail: mario.casu@polito.it).

B. Štitić is with the School of Engineering, Pontificia Universidad Católica de Chile, Santiago, Chile, 7820436 (e-mail: bastitic@uc.cl).

G. Di Guglielmo and L. P. Carloni are with the Department of Computer Science, Columbia University, New York, NY 10027 USA.

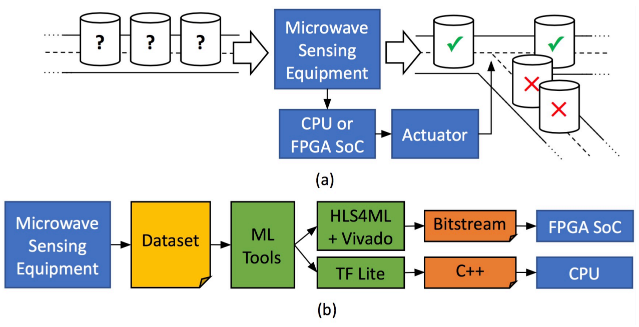

Fig. 1. (a) Microwave (MW) Sensing-based classification in a production line; (b) design flow from MW dataset for Machine-Learning (ML) training to hardware or software implementation of the ML-based MW Sensing classifier.

processing hardware that analyzes the acquired signals and classifies the packaged food as contaminated (positive, red cross mark) or not (negative, green check mark), and activates an actuator to remove the contaminated ones from the line.

Classification of contaminated packaged foods using MW Sensing lends itself quite naturally to Machine Learning (ML). In fact, the abundance of samples to test in production lines can quite easily lead to the generation of datasets on which ML classifiers can be trained. For this reason, in this work we propose an approach called ML-based Microwave Sensing (MLMWS), which was first presented in [9], of which the present work is an extension. We developed the flow depicted in Fig. 1(b), which consists in the following steps:

1) We acquire a dataset of MW measurements obtained in the same operating conditions of a real production line;
2) using ML development tools, we perform a design space exploration of ML classifiers aimed at maximizing a given ML metric (e.g., accuracy, recall, precision) according to the specific industrial requirements;
3) We automatically convert the ML classifier from the ML framework internal description to either a synthesizable hardware specification using *hls4ml* [10], which targets an FPGA System-on-Chip (SoC) implementation, or to a self-contained C++ code to run on a CPU obtained using TensorFlow (TF) Lite [11].
4) Depending on the real-time requirements and the complexity of the ML classifier, we can run the classifier on a CPU or an FPGA SoC, which connects to the MW Sensing equipment and controls the actuator in the production line as shown in Fig. 1(a).

In this paper we apply the flow to a specific industrial case study as described in the remainder of the paper. However, the method can be applied to other industrial scenarios, not only in the food sector but also in other areas in which the packaged products present similar characteristics.

The paper is organized as follows. After a review of the related work in Sec. II, we present the MW system in Sec. III, which also describes the datasets used for training the ML classifiers. Sec. IV is where we present the ML design and the analysis of the performance of the classifiers. The implementation using *hls4ml* and TF Lite is described in Sec. V. After a brief discussion and analysis of perspectives in Sec. VI, we conclude the paper in Sec. VII.

II. Related Work

The renewed interest in ML in the last years has permeated many research fields, including food and agriculture [12]. In this area, ML is most of the times combined with optical systems (e.g., cameras) to detect the presence of contaminants, like insects [13], or to assess the quality, for example in fruits [14]. This last paper has an affinity to our research because it shows a real-time execution of an ML algorithm at the speed of a fast conveyor belt carrying fruits, although the input data is an optical image.

Other relevant examples in the recent literature combine ML with hyperspectral images (from visible to near infrared) of food, as discussed in an extensive review [15]. Mass spectrometry [16] and Raman spectroscopy [17] are also shown to be effectively combined with ML to assess the quality of food against adulteration (in white rice in the cited paper) and to detect food-born pathogens, respectively.

When it comes to exploiting the EM spectrum in combination with ML, a noteworthy example is reported in [18], in which millimeter waves are used for classifying fruits (damaged vs. healthy apples). Microwaves have been used instead of millimeter waves to sense the moisture content in corn in combination with deep neural networks [19]. The sensitivity of microwaves to the moisture content in food has also been exploited in [20], in which the authors describe a hand-held time-domain reflectometer to assess the food quality by measuring variations in the dielectric properties, which can be determined by a variation of water concentration. In this case, however, ML was not used.

MW sensing for food screening using the radar principle has been proposed in [21], although experiments to prove feasibility and accuracy have not been carried out. An industrial implementation of radar-based detection of foreign contaminants, however, is currently commercialized [22]. The radar can detect foreign bodies such as wood, plastic, bone, and fruit stones but is fundamentally different from our approach as it applies to pipes where liquid food or emulsions can flow before being packaged.

Note that none of the above mentioned techniques are shown in the context of packaged foods. A noteworthy exception is shown in [23], in which MW sensing is used to detect plastic contaminants in packaged cheese slices. In this case a single antenna patch is used to illuminate the target and the reflected signal is shown to be sufficiently modified by the contaminant to allow for an easy detection. We will show soon that such an easy condition does not apply to our case study, in which the acquired signals in the contaminated and uncontaminated cases do not exhibit immediately visible patterns useful for a discrimination, hence calling for techniques like ML for a more automated pattern recognition.

III. Microwave Sensing Case Study

In the next two subsections, we explain in detail how the MW Sensing Equipment works, how it can be applied to a specific industrial case, and how the dataset for the ML training was acquired.

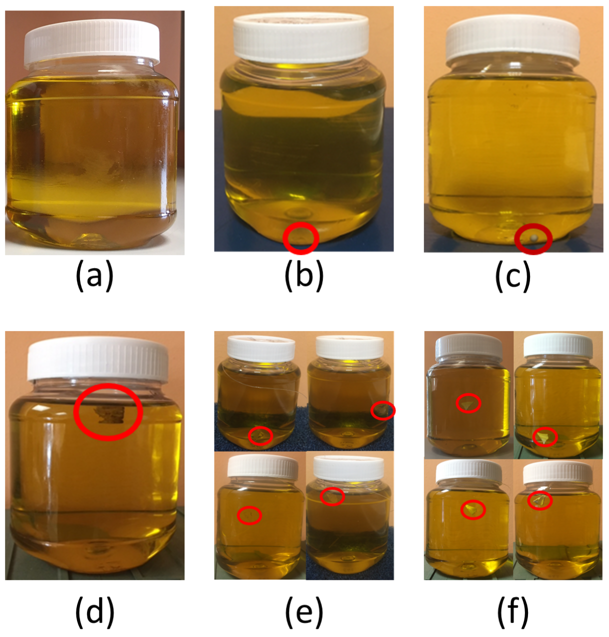

Fig. 2. Samples set, with contaminants circled in red: (a) empty jar; (b) soda-lime glass sample sphere, 2 mm diameter; (c) PTFE sample sphere, 2 mm diameter; (d) fragment of wood, approximately 5 x 5 x 2 mm; (e) small splinter of glass; (f) small splinter of plastic, part of the jar top.

A. MW Sensing System Description

In this case study we conducted our experiments on jars provided by a food company that produces hazelnut-cocoa spread. The jars have a diameter of 6.6 cm and a height of 7.5 cm. The microwave working frequency is centered around 10 GHz, which represents a trade-off between EM waves penetration depth, given the jar dimensions, and the resolution needed to detect millimeter-size intrusions. At such frequency, we measured with a probe the dielectric properties of the food product contained in the jars, and obtained a relative permittivity of 2.86, and a conductivity of 0.21 S/m. Unfortunately, we could not obtain directly contaminated samples from the company, but were given samples of contaminants notoriously difficult to detect with standard methods and industrial devices, as well as standard contaminants used for testing purposes. We were also given instructions on where these contaminants can be typically located in the jar after the packaging.

Therefore, with these contaminants and the related instructions we could recreate the set of contaminated and uncontaminated samples shown in Fig. 2. To facilitate the positioning of the contaminants, we replaced the spread with a transparent oil with the same dielectric characteristics at the frequency of interest, as confirmed by our measurements. The contaminants in the jars are small pieces of plastic or glass, of various shapes and sizes, and wood fragments. According to the instructions, some of these contaminants have just been dropped in the jar, others were placed in various positions within the volume of interest, others were finally left floating at the interface or placed at the bottom of the jar.

We developed an MW sensing system that consists of an array of six antennas mounted on an arch-shaped support (built with a 3D printer) to fit along an industrial line without interrupting the flow of products, and a 6-port Vector Network Analyzer (VNA) [24] connected to the antennas. The number of antennas and their locations have been determined through an accurate numerical analysis of the singular value decomposition of the discretized scattering operator, a method proposed in [25] and used also for medical applications [26], [27].

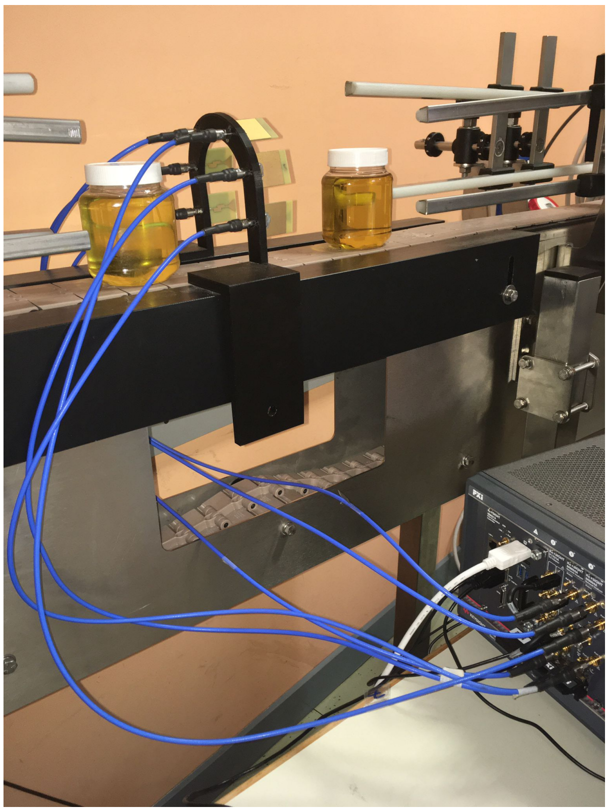

Fig. 3. Microwave sensing system: a 6-port VNA is connected to six monopole antennas mounted on a 3D-printed arch under which the packaged food products can flow uninterruptedly on a conveyor belt.

The picture in Fig. 3 shows the arch of six antennas positioned in a fully-functional industrial conveyor belt available in our laboratory, and also shows the VNA connected to the antennas and the jars moving under the arch. When a jar is about to pass under the arch, a photocell triggers the VNA to start the measurement process. This consists in activating, in turn, one of the six antennas as transmitter while all the antennas (including the transmitting one) capture the scattered electromagnetic field. This results in a 6×6 scattering matrix $S_{i,j}$ that covers the entire volume of the food product under test. Note that compared to our previous work, in which we used a 2-port VNA and an electro-mechanical switching matrix to activate multiple antennas [9], here the 6-port VNA allows us to get rid of the switching matrix and to acquire the measurements with a much faster conveyor belt, hence enabling real-time acquisition and processing in a more realistic industrial scenario. In addition, the measurement is not performed at a single frequency, like in [9], but in the

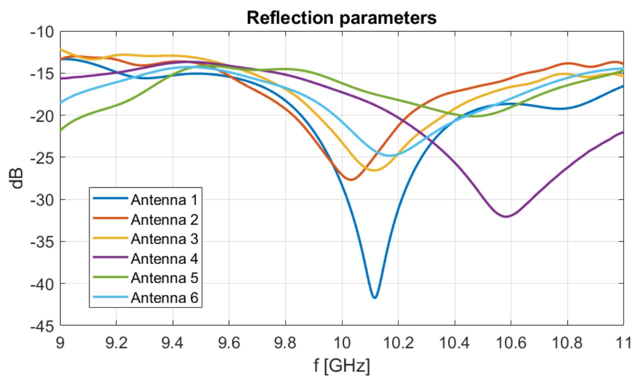

Fig. 4. Reflection parameters of the six antennas used in the microwave sensing system.

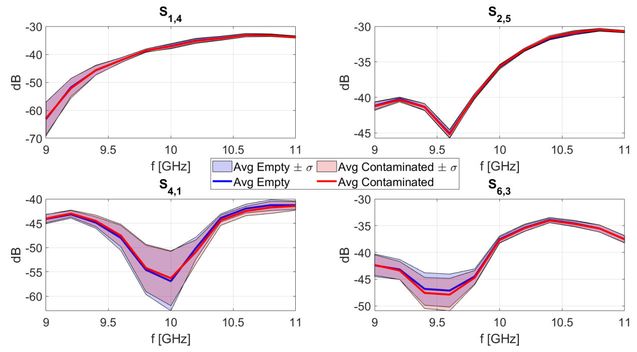

Fig. 5. Magnitude of four scattering parameters measured across the entire dataset of contaminated and uncontaminated samples. The band around the average (thick solid lines) is obtained using the standard deviation.

bandwidth from 9 to 11 GHz with a 200-MHz step.

Despite these improvements, however, the measurement process still takes a non-negligible amount of time, which has two consequences. Firstly, there is less time to perform processing and classification using ML, which makes the design more challenging. Secondly, since the jar moves while the measurements are taken, the resulting scattering matrix is not symmetric as one might expect, due to the fact the $S_{i,j}$ is not measured in the exact same condition of $S_{j,i}$. It turns out that this is actually beneficial, because we obtain more information by "looking" at the target from additional angles that would not be available should the target not move.

The six antennas are low-cost printed monopoles, which have been manually welded to their coaxial feed. This determines a slightly different behavior of the antenna reflection coefficients, as shown in the measurements reported in Fig. 4. Notice that the resonance is not always centered exactly at 10 GHz, which seems to suggest that using information at different frequencies might be a better choice than just using single-frequency measurements.

B. Dataset Creation

We used the microwave sensing system to create a dataset of 1240 samples, of which 600 were free from contaminants and 640 were contaminated with the intrusions described in Fig. 2. We acquired the scattering matrix of each sample in dynamic conditions, with the conveyor belt moving at a speed of 15 cm/s. The scattering matrix is a 6×6 matrix of complex values, which the VNA measures as magnitude and phase, for 11 evenly spaced frequencies from 9 to 11 GHz. For the ML training, the complex values are better represented as real and imaginary couples, hence we converted the matrices from polar to cartesian representation. Moreover, the self-scattering elements of the matrices, i.e., the six diagonal elements, are not used. As a result, we retain, for each of the eleven frequencies available, 60 features per sample, which correspond to the 30 complex numbers of the original matrix that do not lie on the diagonal. In total, $60 \times 11 = 660$ features per sample.

The graphs in Fig. 5 show the magnitude of four matrix elements averaged across the entire dataset partitioned in contaminated and uncontaminated samples as a function of frequency. In addition to the average values (solid thick lines), the graphs also show the band around the average obtained considering the standard deviation of all the measurements in the dataset at any given frequency. We selected these four scattering parameters out of the thirty available for each sample for space reasons, but we observed very similar characteristics in all of them. In particular, we can notice that the family of curves corresponding to contaminated and empty jars completely overlap. This is fundamentally different than what is obtained, for example, in [23], in which the difference between scattering parameters of contaminated and uncontaminated products allow for an immediate classification. Therefore, it is clear that another approach is required, as shown next.

IV. Machine Learning

In this section, we detail the part of the design flow depicted in Fig. 1(b) related to ML. In particular, in the following, we present the design of the Machine Learning (ML) models, their training, and testing results.

A. ML Design

As mentioned earlier, the dataset consists of 1240 samples, of which 640 are contaminated jars, while the remaining 600 are free from intrusions. In what follows, the contaminated samples shall be referred to as the *Positive* class, labeled as '1', whereas the others will correspond to the *Negative* class, labeled as '0'. Similarly to the approach followed in [9], each sample is associated with a NumPy array as required by Python ML development frameworks. Considering the eleven frequencies, each array contains sequences of 330 pairs of real and imaginary parts of each scattering parameter, except for the self-transmission coefficients, which are not considered (i.e., the diagonal elements of the scattering matrix).

We also decided to compare this approach to the creation of the dataset with the approach that we followed in [9], in which we considered only the features measured at the nominal frequency at which the antennas were designed to operate. Therefore, we generated another dataset taking from the previously described one only the features that correspond to the specific frequency of 10 GHz, thus obtaining vectors belonging to a 60-dimensional feature space. By comparing the performance of ML algorithms trained on these two datasets, we hope to gain an insight into whether combining multiple frequencies can improve the detection capabilities or not.

The ML development environment consists of a *Jupyter Notebook* with *Python* 3.7.8 and the following libraries: *Scikit-Learn* 0.23.2 for splitting and preprocessing the datasets, *Scikit-Optimize* 0.8.1 for Bayesian Optimization, *TensorFlow* 2.1.0 and *Keras* 2.4.3 for the MLP model definition, training and predictions.

Both datasets are partitioned into the classical three sets, Training, Validation and Test Set, with Scikit-Learn's `train_test_split`. Training and Validation set are also referred together in the following as the Development Set. The splitting strategy is derived from [28] and is based on the following two key aspects:

1) The size of Test and Validation sets should be large enough to give high confidence on the overall performance attainable with the ML system. Therefore, the first split is between Development and Test sets, which corresponds to a 75%-25% partition (930-310 samples). In the second split, the Development Set is further divided in Training and Validation sets using a 70%-30% proportion (651-279 samples).
2) The Validation and Test sets should ideally contain an equal amount of samples belonging to all classes. For this purpose, stratified sampling, which assigns samples to data groups in a balanced way, is used when deriving the partitions. A significant consequence of this technique is the reduction of the bias of a binary model for one class over the other.

Encouraged by the promising results that we obtained with the Multi-Layer Perceptron (MLP) developed in [9], we decided to continue along this direction and explored various MLP instances in search of the best binary classifier. To efficiently cover the hyperparameter search space of an MLP, we initially adopted the Bayesian Optimization, as we did in [9], through *BayesSearchCV* of the library *Scikit-Optimize*[1].

B. Model Exploration and Training

Initially, we trained the MLP only with the larger dataset (660-dimensional samples). Influenced by the results of our previous work, we did the Bayesian design space exploration in search for relatively complex MLPs with a number of hidden layers between 2 and 6 and neurons per layer between 128 and 1024 (in power-of-two steps). Other hyperparameters were kept fixed, like the sigmoid node for the output layer, the ReLU activation functions for hidden layers, the Binary Cross-Entropy as the loss function, the Adam optimizer with the Keras default learning rate of 0.001 as in [9], the batch size of 256 samples and 500 training epochs.

We realized soon, however, that the Bayesian search returned extremely positive results with excellent accuracy regardless of the number of hidden layers and in general regardless of the *complexity* of the MLP. Without loss of generality, in this work we assume the complexity proportional to the total number of weights of the MLP, as this impacts both computation and memorization aspects of the final implementation. The Bayesian search, however, aims to optimize only the metric chosen for checking the quality of the model on the validation set (e.g., the accuracy), without any consideration for the complexity. A complex MLP indeed can have a significant impact in terms of use of hardware resources (in case the selected platform is an FPGA SoC) and in terms of performance (both for an FPGA SoC and a CPU implementation). This is a very important point for this work, as we aim to meet real-time constraints using dedicated hardware for executing the ML classification.

In any case, this initial automated search gave us hints about the potential of low-complexity MLPs in providing an excellent level of accuracy. This result came as a surprise because we were expecting to need an MLP of comparable complexity to the one used in [9]. Most likely, the improvement granted by the microwave sensing tools used in this case and the removal of the electro-mechanical switching matrix helped "declutter" the signals from various artifacts and noise, ultimately leading to the simplification of the ML classifier.

For this reason, guided by the hints provided by the results of the initial automated search, we started a manual exhaustive search aimed at MLPs of limited complexity. Thus, for both datasets, we considered MLPs with only 1 hidden layer and a number of nodes/neurons equal to 1, 2, 4, 8, 16, 32 or 64. Therefore, in total, we trained fourteen models. No dropout was required since the classifiers rarely showed overfitting. In addition, the optimizer, the learning rate and the batch size remained as previously indicated for the Bayesian search. We trained all the models on the 660-dimensional dataset for 500 epochs, while for those trained with the 60-dimensional samples we used 2000 epochs. Finally, we kept the weights of the last epoch, because we always observed a converging behavior in the curves of validation loss, without any oscillation.

It should be emphasized that, before starting the manual training, we always performed preprocessing. To decide which technique to use, the ranges of all the features were explored for each dataset. It was observed that values for a given feature could differ by up to 5 orders of magnitude. Furthermore, no atypical values were detected. As a result, scaling was considered, particularly standardization. Having already shown promising results in [9], a StandardScaler object was fit separately for each Training Set of the two datasets.

C. Model Selection

We show the fourteen trained models as scattered points in Fig. 6. In detail, Fig. 6(a) reports the value of the loss function on the Validation Set, as obtained from the last and best training epoch, as a function of the complexity; Fig. 6(b) shows the corresponding validation accuracy. In the key, N indicates the number of hidden neurons and is strictly related to the number of weights (complexity); A (squares) stands for classifiers trained on the dataset with eleven frequencies; B (circles) refers to those trained on the 10-GHz dataset.

Notice that while the validation accuracy is very high for all the classifiers, the variation of the validation loss can be

[1]See https://github.com/scikit-optimize/scikit-optimize for details on this library for the optimization of noisy black-box functions.

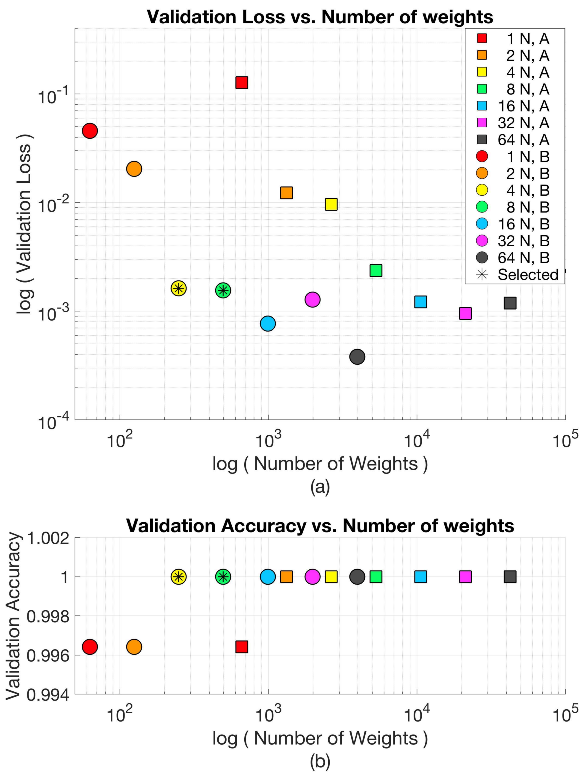

Fig. 6. Trained classifiers in the plane of complexity (i.e., number of weights) versus validation loss in (a) (log scale on both axes) and versus validation accuracy in (b) (log scale on the x-axis only). N is the number of neurons; A is for the models trained on the full dataset with eleven frequencies; B is for those trained on the single-frequency dataset.

an indication of a difference in the training outcome: the classifiers with a lower loss should have learned better than those with a greater loss and therefore should generalize better when applied to unseen samples. The results of the training shows that the validation loss decreases as the complexity of the model increases, which is often the case in ML up until the higher complexity leads to overfitting, which we did not observe in our training experiments. The results also show that, contrary to our initial hypothesis, the models trained on the dataset with the 660 features obtained with all the frequencies (A) do not necessarily perform better than those trained on the dataset with 60 features measured at 10 GHz (B). As a matter of fact, interpreting Fig. 6(a) as a Pareto plot, the squares are dominated by the circles.

Based on these outcomes, we decided to consider only the models trained with the dataset obtained at 10 GHz as potential candidates for the implementation. To choose the final model, we considered the confusion matrix obtained on the Validation Set. This is defined as

$$\begin{pmatrix} TN & FP \\ FN & TP \end{pmatrix},$$

where *TN* are the *True Negatives*, *TP* the *True Positives*, *FN* the *False Negatives*, and *FP* the *False Positives*. An *ideal*

TABLE I
CONFUSION MATRICES OF THE SEVEN B MODELS CALCULATED ON THE VALIDATION SET.

Model Name	**Confusion Matrix**	**Confusion Matrix Normalized**
1 N, B	$\begin{pmatrix} 135 & 0 \\ 1 & 143 \end{pmatrix}$	$\begin{pmatrix} 1.0 & 0.0 \\ 0.007 & 0.993 \end{pmatrix}$
2 N, B	$\begin{pmatrix} 135 & 0 \\ 1 & 143 \end{pmatrix}$	$\begin{pmatrix} 1.0 & 0.0 \\ 0.007 & 0.993 \end{pmatrix}$
4 N, B	$\begin{pmatrix} 135 & 0 \\ 0 & 144 \end{pmatrix}$	$\begin{pmatrix} 1.0 & 0.0 \\ 0.0 & 1.0 \end{pmatrix}$
8 N, B	$\begin{pmatrix} 135 & 0 \\ 0 & 144 \end{pmatrix}$	$\begin{pmatrix} 1.0 & 0.0 \\ 0.0 & 1.0 \end{pmatrix}$
16 N, B	$\begin{pmatrix} 135 & 0 \\ 0 & 144 \end{pmatrix}$	$\begin{pmatrix} 1.0 & 0.0 \\ 0.0 & 1.0 \end{pmatrix}$
32 N, B	$\begin{pmatrix} 135 & 0 \\ 0 & 144 \end{pmatrix}$	$\begin{pmatrix} 1.0 & 0.0 \\ 0.0 & 1.0 \end{pmatrix}$
64 N, B	$\begin{pmatrix} 135 & 0 \\ 0 & 144 \end{pmatrix}$	$\begin{pmatrix} 1.0 & 0.0 \\ 0.0 & 1.0 \end{pmatrix}$

classifier would have *FN* $= 0$, *FP* $= 0$, and nonzero values only on the diagonal of the confusion matrix.

Tab. I shows that all the seven models are very good classifiers, even though the models with four or more neurons are actually ideal classifiers, at least in the validation set. Notice that the errors of the models with one or two neurons are only false negatives. For some applications, *FP* and *FN* errors are not interchangeable, as one kind of error can have more serious consequences than the other. This is exactly the case for this application, since false negatives mean that some contaminated jars are undetected, whereas false positives mean that some uncontaminated jars are discarded. In general, dropping a few uncontaminated items is considered more acceptable by food companies, as long as the number is limited and does not have a sizeable economic impact, than letting even fewer contaminated items reach the market. For this reason, models $1N, B$ and $2N, B$ were abandoned.

Among the other ones, we selected the first two models, $4N, B$ and $8N, B$, which correspond to the points indicated with a $*$ in Fig. 6. We kept both because the first one is less complex (249 vs. 497 number of weights), while the second one has a slightly lower validation loss (1.62e-03 vs. 1.55e-03), with the same validation accuracy (equal to 1.000), as shown in Fig. 6. Moreover, we considered that the Validation Set does not contain enough samples to allows us to select one of the two with the highest statistical confidence. Therefore, in the following we present results obtained on the Test Set for both of these cases. Also, these two models will be compared in terms of performance and resource utilization in Sec. V.

D. Model Testing

The evaluation of the predictive performance of models $4N, B$ and $8N, B$ is reported in Tab. II. The confusion matrices were computed on the Test Set obtained with the splitting of the dataset previously described, as well as on a New Test Set. In fact, to further assess the generalization capabilities of

TABLE II
CONFUSION MATRICES OF THE SELECTED MODELS CALCULATED ON THE VALIDATION, TEST AND NEW TEST SETS.

Model Name	**Confusion Matrix Val Set**	**Confusion Matrix Val Set Normalized**	**Confusion Matrix Test Set**	**Confusion Matrix Test Set Normalized**	**Confusion Matrix New Test Set**	**Confusion Matrix New Test Normalized**
4 N, B	$\begin{pmatrix}135 & 0\\ 0 & 144\end{pmatrix}$	$\begin{pmatrix}1.0 & 0.0\\ 0.0 & 1.0\end{pmatrix}$	$\begin{pmatrix}148 & 2\\ 0 & 160\end{pmatrix}$	$\begin{pmatrix}0.987 & 0.013\\ 0.0 & 1.0\end{pmatrix}$	$\begin{pmatrix}148 & 2\\ 0 & 260\end{pmatrix}$	$\begin{pmatrix}0.987 & 0.013\\ 0.0 & 1.0\end{pmatrix}$
8 N, B	$\begin{pmatrix}135 & 0\\ 0 & 144\end{pmatrix}$	$\begin{pmatrix}1.0 & 0.0\\ 0.0 & 1.0\end{pmatrix}$	$\begin{pmatrix}150 & 0\\ 0 & 160\end{pmatrix}$	$\begin{pmatrix}1.0 & 0.0\\ 0.0 & 1.0\end{pmatrix}$	$\begin{pmatrix}150 & 0\\ 0 & 260\end{pmatrix}$	$\begin{pmatrix}1.0 & 0.0\\ 0.0 & 1.0\end{pmatrix}$

the two selected classifiers, 100 additional positive samples were acquired with different contaminants than those used in the former dataset. These additional intrusions are a nylon sphere (2 mm radius), a new glass fragment (8 mm long side, 2 mm thick) and a new type of plastic (10 mm long side, few millimeters thick). The features of the New Test Set were obtained with the same Microwave Sensing Equipment and were later standardized using the mean and standard deviation metrics which were also applied to the Test Set.

The confusion matrix obtained on the Validation Set, reported earlier in Tab. I, is also copied in Tab. II to allow for a better comparison with the predictions obtained on the two test sets. We can observe a slight performance degradation in the Initial Test Set for model $4N, B$, with two *FP* errors, which corresponds to a test accuracy of 99.35%. Notice that the New Test Set does not introduce any additional errors. Based on the previous consideration about the difference between *FP* and *FN* errors for the hazelnut-cocoa spread application, we conclude that the performance of the $4N, B$ is acceptable, as it does not show any *FN* misprediction. On the other hand, model $8N, B$ correctly classifies all test samples.

In summary, we decided to implement the two selected models for the following reasons:

- Model $4N, B$ has half the complexity of $8N, B$ (249 vs. 497 weights) while still able to reach the condition *FN* = 0, which is very important for this application.
- Model $8N, B$, on the other hand, is the best model in terms of obtained accuracy on the Test and New Test sets, even if it has a larger complexity.

As the final step in our ML flow, we exported the two MLPs in the proper format required by hls4ml [29]: the information about the architecture in JavaScript Object Notation (JSON) and the weights in Hierarchical Data Format (H5). Moreover, the two Keras models are converted to TF Lite models in order to implement them on an ARM-based CPU. This step is described in V-B.

V. IMPLEMENTATION

As outlined in Sec. I, after the training and the selection of the best models, we can target two implementations on two different platforms aimed at systems with different requirements and models with different complexity. For this specific case study, we verified that both platforms are compatible with the real-time requirements of the application and the selected models, as shown in the remainder of this section.

A. FPGA SoC Platform Flow Using hls4ml

In modern embedded systems, heterogeneous computing has become predominant, with a workload split between nodes working concurrently such as CPU, FPGA, and sometimes also ASIC [30]. Current FPGA SoCs, in particular, offer in one small form-factor system the simplicity and ease of programming of a CPU, termed Processing System (PS) in Xilinx devices, coupled with the power of spatially parallel and temporally pipelined processing of an FPGA, which is the Programmable Logic (PL) in Xilinx terminology.

Traditionally, software or hardware designers adopt the most appropriate programming language for each heterogeneous system component. In particular, hardware designers utilize hardware-description languages such as Verilog or VHDL for FPGA. In recent years, high-level synthesis (HLS) has become an alternative for generating hardware modules from code written in programming languages such as C/C++ or SystemC [31]. HLS comes with significant benefits: it raises the level of abstraction and reduces the simulation speed; it simplifies the validation phases; and finally, it makes the exploration and evaluation of design alternatives easier.

Although HLS-based approaches are more accessible for domain experts than the HDL-based ones, the expert still needs to be aware of a variety of HLS directives (or knobs) and how they affect cost and performance of the final system. HLS code written without understanding the inferred hardware often yields results worse than merely running the corresponding software implementation on a CPU. Hence, the domain expert must carefully combine a synthesizable code with directives to generate efficient hardware [32]. This manual process significantly impacts the overall system development, and this situation becomes more acute for complex ML applications that share a base of standard functionalities, like dense, convolutional, activation, and pooling layers. Domain experts rarely have the expertise or time to undertake these challenges.

In our context, the domain expert is a hypothetical microwave engineer with ML skills, who needs to accelerate a given ML algorithm, like the classifiers here at stake, on an FPGA SoC platform. To this domain expert, our workflow offers a method to automatically generate efficient synthesizable C++ code and HLS directives from the trained Keras/TensorFlow model. This method is based on the *hls4ml* framework [29].

This framework first compiles the model to an intermediate representation (IR); then it maps the IR on state-of-the-art implementations of ML layers and combines them with

TABLE III
HLS RESOURCE ESTIMATIONS.

Model Name	Period (ns)	Lat. (clk #)	II (clk #)	BRAM (%)	DSP (%)	FF (%)	LUT (%)
4 N, B	10	26	16	1	14	5	11
8 N, B	10	27	16	1	28	6	21

the correct HLS directives to extract parallelism; finally, the generated code is fed to an HLS tool to generate an RTL IP for the programmable logic. *hls4ml* provides a Python API that allows the design automation of ML devices by just using a high-level knowledge of the final hardware and a minimal set of hardware design knobs. For example, besides target clock period and the fixed-point precision, *hls4ml* introduces the concept of *reuse factor*, as a single configuration knob to specify the operation parallelism in the layers of neurons.

The parallelization of the inference calculation and, in particular, the number of multiplications performed in parallel determines the trade off between latency, throughput and FPGA resource usage. In *hls4ml*, the reuse factor sets the number of times a multiplier is used in the computation of a layer output values. With a reuse factor of one, the layer computation is fully parallel, i.e. each multiplier is used once and the Initiation Interval (II) is one, which means that a new computation can start at each clock cycle. With a reuse factor of R, $1/R$ of the computation is done at a time with a factor of $1/R$ fewer multipliers and the layer II is $1/R$. With this knob, the designer can quickly explore the design space in search of the solution that achieves the best trade-off between performance and utilization of FPGA resources.

Tab. III reports Xilinx Vivado HLS resource estimations for the two models described in Sec. IV when targeting a Xilinx Zynq-7000 SoC (`xc7z020-1clg400c`) that combines Artix-7 programmable logic with a dual-core ARM Cortex-A9 processor at 650 MHz. The target clock period is 10 ns, the fixed-point precision is 16 bits for the word size and 6 bits for the signed integer part (`fixed<16,6>`), and the reuse factor for all of the layers is 16.

Fig. 7 shows the integration of the *hls4ml*-generated IP on the target Zynq system. The accelerator sits in the PL part; the ARM core and DDR3 main memory are in the PS one. Data is transferred between PS and PL through high performance (HP) AXI slave ports, while control signals generated by the PS is transferred through a general purpose (GP) AXI-Lite master port. For these experiments, we extended the *hls4ml*-generated IP with AXI interfaces and local buffer. The AXI datamovers read the features and return the inference results in DDR3 memory. The AXI-Lite port is used for the ARM CPU to program the accelerator configuration and control the accelerator execution. The accelerator configurations include physical start addresses of the input features and output predictions. The PL is driven by single clock domain of 100 MHz generated by the PS.

Tab. IV shows the performance and final resource utilization for the whole PL after synthesis and implementation. Additional BRAMs and LUTRAMs are used to implement

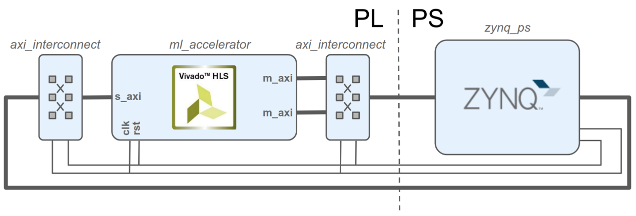

Fig. 7. System setup.

TABLE IV
FINAL SYSTEM RESOURCES. FREQUENCY AND POWER ARE 100 MHz AND 1.8 W, RESPECTIVELY, FOR BOTH MODELS.

Model Name	Lat. (μs)	BRAM (%)	DSP (%)	FF (%)	LUT (%)	LUTRAM (%)
4 N, B	2	1	14	11	33	1
8 N, B	2	1	28	11	39	1

the local buffer associated with the AXI datamovers. As in the HLS estimates, we can notice that the DSPs are critical resources and approximately double when we transition from the 4-neuron model to the 8-neuron model. We also report the total power for the board measured by a USB power meter.

The latency column shows that the timing performance for the inference is only $2\,\mu$s on this hardware platform. This is clearly negligible compared to the measurement time and the time to transfer the sixty features from the microwave equipment to the FPGA. Thus, we can conclude that this solution is perfectly compatible with the real-time requirements of this and similar applications.

B. CPU Flow Using TF Lite

To ease the process of converting trained ML models from a TensorFlow (or Keras) development environment so as to run them on embedded devices, *TensorFlow Lite* (TF Lite) [11] comes into play. TF Lite targets mobile, embedded, and IoT devices, as opposed to the standard TF used in the cloud or desktops. TF Lite aims at ML inference on simpler devices with less computing capabilities, while targeting a low latency and a small binary size [11]. It consists of two main components. The first is the *interpreter,* which understands the operations that the ML model needs to perform, such as convolutions, dense layers, etc., and then it uses at run-time the proper optimized library for the specific operation, considering also the target hardware. The second component is the *converter,* which elaborates and optimizes the TF models at compile-time to make them compatible for the interpreter and, at the same time, to reduce the binary size and to improve the performance.

In this work we decided to use a more recent TF branch, called *TensorFlow Lite Micro* (TF Micro). According to the developers, TF Micro is an open-source ML inference framework for running deep-learning models on embedded systems [33], in particular microcontroller/embedded CPUs. TF Micro introduces a special, space-efficient format to encode the model and its parameters to be used in memory-constrained devices in an optimized way. This is particularly useful for the numerous microcontroller platforms without native filesystem support. For these platforms, TF Micro offers the possibility to load the entire model as a single C array and compile it into the program that requires the model. We took this path because it clearly facilitates the inclusion of the model in the main code that calls the inference routines.

One interesting option offered by TF Lite, is the ability to convert the model parameters and internal variables from floating-point in 32 bits (FP32) to 8-bit integers (INT8). This is typically convenient for platforms with a limited amount of on-chip SRAMs (normally, in the order of a few kilobytes), as well as for those simple processors without FP units. TF Lite performs a post-training quantization procedure aimed at determining the best way to perform the type conversion that minimally affects the accuracy. In our case, the complexity of the selected MLP models was already sufficiently low so that the conversion was not strictly necessary. Moreover, we considered as target CPU the dual-core ARM Cortex-9 available on the PS of the Xilinx Zynq-7000 SoC, which indeed possesses an FP unit. Nonetheless, we decided to compare the original FP32 model to the quantized INT8 one in terms of accuracy and latency, so as to evaluate the potential of this quantization technique.

Two steps are required to run the inference on the ARM cores of a Zynq SoC. First, we need to cross-compile the TensorFlow Lite Micro library for the target ARM Cortex-A9 (processor) core (`libtensorflow-microlite.a`). Then, we need to create a C++ application in Xilinx Vivado SDK by leveraging the examples provided with the codebase. The application instantiates the model that it has encoded as a char array in a header file (`model.h`), passes the input values, runs the model (`tflite::MicroInterpreter::Invoke()`), and checks for correctness of the predictions. It is important to remember that we had some lines of code to preprocess the input samples with standardization. We profiled the performance of the two selected MLPs, each in the FP32 and INT8 variants. In terms of accuracy we did not observe any change from the results presented in Sec. IV, which means that TF Lite does a very good job at quantizing from FP32 to INT8. The inference performance results are reported in Tab. V. The second column reports the type of data representation, either `float32` or `full INT8`. The fourth and fifth columns report the latency results in microseconds and the On-Chip Memory footprint of the four models in Bytes. This is the size of the C source file generated by TF Micro, which in turns comprises the description of the model architecture and the weights. The frequency of the processor was always set to 650 MHz.

From Tab. V one can see that all the weights and activations of the four models are way below the limit of 256 kB of the On-Chip memory of the target processor. This means that during the inference, all the partial output activations of the fully connected hidden layer are kept in memory and no DDR transfers are needed to store them temporarily.

Regarding the latency results, these are more or less 10x greater than those obtained in the FPGA SoC implementation described in Sec. V.A. As expected, the latency of the models with four neurons is less than half the latency with eight neurons. Somewhat unexpectedly, the latency of the INT8 versions are greater than the FP32 counterparts. This is due to the overhead of the quantization and dequantization phases performed at the input and output of the MLP network as well as in the internal operations [34].

TABLE V
TF LITE MICRO IMPLEMENTATION RESULTS ON THE ARM CORTEX-A9 PROCESSOR.

Model Name	Type of compression	Freq. (MHz)	Lat. (us #)	On-Chip Mem. (Bytes #)
4 N, B	float32	650	8	2380
4 N, B	full INT8	650	19	2240
8 N, B	float32	650	10	3380
8 N, B	full INT8	650	22	2504

VI. DISCUSSION AND PERSPECTIVES

In this section, we would like to offer an open discussion on the outcome of our research, hoping that this would help other researchers and practitioners in this field and possibly attract the attention and interest of others. We will divide the discussion into three parts. The first will be about the interrelation between microwave sensing and ML, the second on the hardware and software implementation of the ML classifiers, and the third on the perspectives and future works.

A. Microwave Sensing and ML Interrelation

In terms of classification accuracy and sensitivity, the results obtained with our approach on the Test Set are surprisingly good. The new microwave equipment and dataset allow us to obtain major improvements upon the results of our previous experiments [9], which were already very positive. This is the very reason why we decided to further test the generalization performance of the selected models with the New Test Set, as discussed in Sec. IV-D. In addition to this experiment on the New Test Set, we performed two simple additional experiments, both successful, as sanity checks:

1) We changed the splitting policy by reducing the percentage of samples in the Training Set while increasing progressively the size of the Validation and Test sets. Then, we trained the previously selected models ($4N, B$ and $8N, B$) from scratch with each new train-val-test split. We noticed that the classifiers continued to perform very well on the Test Set until the number of training samples fell below a threshold of 260 samples. In this case, the models showed underfitting behavior and were unable to generalize as before. These results suggest that our choice of splitting the dataset into 75% for development and 25% for testing was reasonably correct and was not the cause of the high accuracy we obtained.
2) We swapped all the nylon samples, which were only present in the New Test Set (20 samples), with all the wood samples, which were only included in the initial

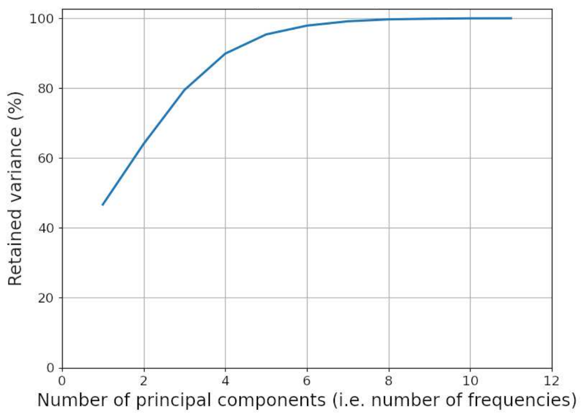

Fig. 8. Retained variance as a function of the number of principal components for the multi-frequency dataset.

dataset (100 samples). Despite the fact that a) wood has significantly different electromagnetic properties than the other materials, and b) we replaced many more wood samples with much fewer nylon samples, the results were similar to those reported in Sec. IV-D. Achieving good performance metrics after such swapping experiment confirms that the high quality of the results is not determined by a fortunate selection of the contaminants.

We ascribe these results primarily to the quality of the new equipment, although supplemental investigation might be required to confirm this hypothesis. Interestingly, additional frequencies do not seem to bring about any improvement compared to a single-frequency dataset. This also came as a surprise, given that a principal component analysis of the dataset aimed at determining the number of independent "frequencies" reveals that at least 5 to 6 components are needed for a retained variance of 90%, as shown in Fig. 8. Note that in some experiments, not reported in this paper and performed with the equipment used in our original contribution [9], we observed an accuracy that tended to increase with the number of frequencies included in the dataset. Even though the fact that a single frequency is sufficient for a perfect classification is certainly welcome as it can speed up or simplify the microwave measurements, still it would be interesting to have a deeper insight as to why this happens.

B. Hardware and Software Implementation

Regarding the hardware and software implementation of the classifiers on the target platforms, we obtained negligible latency for both the CPU and the FPGA. This might lead to conclude that an FPGA implementation is unnecessary. Although this might be true for the particular case study that we considered, this is not necessarily true in general. For the previously mentioned case, not reported here, in which more frequencies led to better accuracy, the best classifiers had a higher complexity that made the software implementation inevitably slow. In addition, there are practical industrial cases where the speed of the conveyor belt is up to ten times faster than what we considered (for example, in the beverage industry). All things considered, we believe that having a path from dataset acquisition to deployment in either an accelerated hardware implementation on FPGA for those specific cases, as well as a standard embedded software solution for all the other cases is definitely an advantage of our methodology.

C. Perspectives and Future Works

Finally, we plan to apply the method and technique described in this paper to other food industrial cases, as well as to other industrial sectors. We believe that any time a typically homogeneous target is affected by a low-density contaminant, this would be a perfect case for ML-based microwave sensing. As an example, some pharmaceutical products (e.g., bottled liquids) belong to this broad category. The adaptation would require a system refinement, depending on the dielectric features of the product to inspect and the shape and relative dimensions of its container. The architecture and the operating frequency must be set to allow a sufficient penetration depth inside the product, depending on its permittivity and conductivity. As an example, liquid products, which consist mainly of water, have totally different properties from the cream used in this work in terms of signal absorption, which is much stronger in aqueous media. For this reason, a lower frequency would be needed to penetrate the liquid. On the other hand, the dielectric contrast between the liquid product and a low-density contaminant will be greater, making the contamination easier to be located compared to the oil-based products considered in our work. As a matter of fact, the dielectric contrast between plastic and water is more than 20x greater than that between cream and plastic.

Apart from the mentioned intrinsic limitations of microwaves, which cannot penetrate metallic packaging, another constraint to consider is that the signal scattered by an intrusion must be above the noise floor. The signal dynamic of the sensing system is consequently a further key aspect to consider when adapting such system to different media.

VII. Conclusion

In this paper we presented a case study of a Machine Learning-based Microwave Sensing approach to the detection of contaminants in packaged food. We built upon our previous research presented in [9] and reported results obtained using an improved microwave sensing system, which allowed us to acquire and process microwave data in real-time from a real industrial production line with a moving conveyor belt carrying food jars. With a dataset of jar samples contaminated with various intrusions typically found in a production environment as well as uncontaminated samples, we performed a design space exploration aimed at selecting the best MLP binary classifiers trained on the dataset. We managed to obtain both exceptionally accurate and light-weight models, for which we present a complete flow from training to final deployment. Based on the requirements of the application and the characteristics of the model, with our flow we can target the most appropriate platform for running the ML algorithms in real-time, including an FPGA SoC when the speed requirements call for it, or an ARM-based CPU when such requirements are less strict.

Even though the approach proposed in this paper has been shown to be effective for a specific case study, we are confident that it can be applied to other scenarios with similar characteristics, both in the food sector and other industrial sectors, especially when contaminants that escape the standard methods are to be detected, like low-density plastics.

Acknowledgment

The authors would like to thank the Developer Team of *hls4ml*. They would also acknowledge the Fast Machine Learning collective as an open community of multi-domain experts and collaborators. This community was important for the development of this project.

References

[1] The rapid alert system for food and feed. [Online]. Available: https://ec.europa.eu/food/sites/food/files/safety/docs/rasff_annual_report_\2018.pdf

[2] R. Haff and N. Toyofuku, "X-ray detection of defects and contaminants in the food industry," *Sensing and Instrumentation for Food Quality and Safety*, vol. 2, Dec. 2008.

[3] N. Lorenzi. (2019, January) Inspection and detection equipment reach new levels of flexibility for peak food safety. [Online]. Available: https://www.foodsafetystrategies.com/articles/613-inspection-and-detection-equipment-reach-new-levels-of-flexibility-for-peak-food-safety

[4] W.-H. Lee and W. Lee, "Food inspection system using terahertz imaging," *Microwave and Optical Technology Letters*, vol. 56, no. 5, pp. 1211–1214, 2014.

[5] K. Wang, D.-W. Sun, and H. Pu, "Emerging non-destructive terahertz spectroscopy imaging technique: Principle and applications in the agri-food industry," *Trends in Food Science & Technology*, vol. 67, pp. 93–105, 2017.

[6] C. Jordens and M. Kock, "Detection of foreign bodies in chocolate with pulsed THz spectroscopy," *Optical Engineering*, vol. 47, no. 3, pp. 93–105, 2008.

[7] Z. Wu and H. Wang, "Microwave tomography for industrial process imaging: Example applications and experimental results," *IEEE Antennas and Propagation Magazine*, vol. 59, no. 5, pp. 61–71, 2017.

[8] J. LoVetri, M. Asefi, C. Gilmore, and I. Jeffrey, "Innovations in electromagnetic imaging technology: The stored-grain-monitoring case," *IEEE Antennas and Propagation Magazine*, vol. 62, no. 5, pp. 33–42, 2020.

[9] L. Urbinati, M. Ricci, G. Turvani, J. A. T. Vasquez, F. Vipiana, and M. R. Casu, "A machine-learning based microwave sensing approach to food contaminant detection," in *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2020, pp. 1–5.

[10] J. Duarte, S. Han, P. Harris, S. Jindariani, E. Kreinar, B. Kreis, J. Ngadiuba, M. Pierini, R. Rivera, N. Tran *et al.*, "Fast inference of deep neural networks in FPGAs for particle physics," *Journal of Instrumentation*, vol. 13, no. 07, p. P07027, 2018.

[11] (2021) TensorFlow Lite Guide | TensorFlow. [Online]. Available: https://www.tensorflow.org/lite/guide

[12] L. Zhou, C. Zhang, F. Liu, Z. Qiu, and Y. He, "Application of deep learning in food: A review," *Comprehensive Reviews in Food Science and Food Safety*, vol. 18, no. 6, pp. 1793–1811, 2019.

[13] H. Bisgin, T. Bera, H. Ding, H. Semey, L. Wu, Z. Liu, A. Barnes, D. Langley, M. Pava-Ripoll, H. Vyas, W. Tong, and J. Xu, "Comparing svm and ann based machine learning methods for species identification of food contaminating beetles," *Scientific Reports*, vol. 8, no. 6532, 2018.

[14] S. Fan, J. Li, Y. Zhang, X. Tian, Q. Wang, X. He, C. Zhang, and W. Huang, "On line detection of defective apples using computer vision system combined with deep learning methods," *Journal of Food Engineering*, vol. 286, p. 110102, 2020.

[15] R. Lankapalli, D. Jayas, N. White, P. Fields, and D.-W. Sun, "Extraction of spectral information from hyperspectral data and application of hyperspectral imaging for food and agricultural products," *Food Bioprocess Technology*, vol. 10, pp. 1–33, 2017.

[16] D. K. Lim, N. P. Long, C. Mo, Z. Dong, L. Cui, G. Kim, and S. W. Kwon, "Combination of mass spectrometry-based targeted lipidomics and supervised machine learning algorithms in detecting adulterated admixtures of white rice," *Food Research International*, vol. 100, pp. 814–821, 2017.

[17] S. Yan, S. Wang, J. Qiu, M. Li, D. Li, D. Xu, D. Li, and Q. Liu, "Raman spectroscopy combined with machine learning for rapid detection of food-borne pathogens at the single-cell level," *Talanta*, vol. 226, p. 122195, 2021.

[18] F. Zidane, J. Lanteri, L. Brochier, N. Joachimowicz, H. Roussel, and C. Migliaccio, "Damaged apple sorting with mmwave imaging and nonlinear support vector machine," *IEEE Transactions on Antennas and Propagation*, vol. 68, no. 12, pp. 8062–8071, 2020.

[19] J. Zhang, D. Du, Y. Bao, J. Wang, and Z. Wei, "Development of multifrequency-swept microwave sensing system for moisture measurement of sweet corn with deep neural network," *IEEE Transactions on Instrumentation and Measurement*, vol. 69, no. 9, pp. 6446–6454, 2020.

[20] O. Schimmer, F. Daschner, and R. Knochel, "Uwb-sensors in food quality management — the way from the concept to market," in *2008 IEEE International Conference on Ultra-Wideband*, vol. 2, 2008, pp. 141–144.

[21] M. Edwards, *Detecting foreign bodies in food*. Woodhead Publishing Ltd, 2004.

[22] [Online]. Available: http://www.foodradar.com/

[23] M. Muradov, P. Kot, M. Ateeq, B. Abdullah, A. Shaw, K. Hashim, and A. Al-Shamma'a, "Real-time detection of plastic shards in cheese using microwave-sensing technique," *Proceedings of The 6th International Electronic Conference on Sensors and Applications*, vol. 42, no. 1, 2020.

[24] Keysight Technologies, "M980xA Series PXIe Vector Network Analyzer," *Data Sheet*, 2020.

[25] R. Scapaticci, J. A. Tobon Vasquez, G. Bellizzi, F. Vipiana, and L. Crocco, "Design and numerical characterization of a low-complexity microwave device for brain stroke monitoring," *IEEE Trans. Antennas Propag.*, vol. 66, pp. 7328–7338, Dec. 2018.

[26] Jorge A. Tobon Vasquez et al., "Design and experimental assessment of a 2D microwave imaging system for brain stroke monitoring," *Int. J. Antennas Propag.*, no. Article ID 8065036, p. 12 pages, 2019.

[27] J. A. Tobon Vasquez, R. Scapaticci, G. Turvani, M. Ricci, L. Farina, A. Litman, M. R. Casu, L. Crocco, and F. Vipiana, "Noninvasive inline food inspection via microwave imaging technology: An application example in the food industry," *IEEE Antennas and Propagation Magazine*, vol. 62, no. 5, pp. 18–32, 2020.

[28] A. Ng, "Structuring Machine Learning Projects." [Online]. Available: https://www.coursera.org/learn/machine-learning-projects

[29] F. Fahim, B. Hawks, C. Herwig, J. Hirschauer, S. Jindariani, N. Tran, L. P. Carloni, G. D. Guglielmo, P. Harris, J. Krupa, D. Rankin, M. B. Valentin, J. Hester, Y. Luo, J. Mamish, S. Orgrenci-Memik, T. Aarestaad, H. Javed, V. Loncar, M. Pierini, A. A. Pol, S. Summers, J. Duarte, S. Hauck, S.-C. Hsu, J. Ngadiuba, M. Liu, D. Hoang, E. Kreinar, and Z. Wu, "hls4ml: An open-source codesign workflow to empower scientific low-power machine learning devices," 2021.

[30] L. P. Carloni, "The case for embedded scalable platforms," in *Proc. the 53rd Annual Design Automation Conference (DAC)*. New York, NY, USA: Association for Computing Machinery, 2016.

[31] S. Lahti, P. Sjövall, J. Vanne, and T. D. Hämäläinen, "Are we there yet? A study on the state of high-level synthesis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 38, no. 5, pp. 898–911, 2018.

[32] L. Piccolboni, P. Mantovani, G. D. Guglielmo, and L. P. Carloni, "Cosmos: Coordination of high-level synthesis and memory optimization for hardware accelerators," *ACM Trans. Embed. Comput. Syst.*, vol. 16, no. 5s, Sep. 2017. [Online]. Available: https://doi.org/10.1145/3126566

[33] R. David, J. Duke, A. Jain, V. J. Reddi, N. Jeffries, J. Li, N. Kreeger, I. Nappier, M. Natraj, S. Regev, R. Rhodes, T. Wang, and P. Warden, "Tensorflow lite micro: Embedded machine learning on tinyml systems," 2020.

[34] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2704–2713.