Enhanced Machine-Learning Flow for Microwave-Sensing Systems to Detect Contaminants in Food

 Full citation of the published work: B. Štitić, L. Urbinati, G. Di Guglielmo, L. Carloni and M. R. Casu, "Enhanced Machine-Learning Flow for Microwave-Sensing Systems to Detect Contaminants in Food," 2023 IEEE Conference on AgriFood Electronics (CAFE), Torino, Italy, 2023, pp. 40-44, doi: 10.1109/CAFE58535.2023.10291198.

Enhanced Machine-Learning Flow for Microwave-Sensing Systems to Detect Contaminants in Food

Bernardita Štitić, Luca Urbinati, Giuseppe Di Guglielmo, Luca Carloni, Mario R. Casu
bastitic@uc.cl, luca.urbinati@polito.it, gdg@fnal.gov, luca@cs.columbia.edu, mario.casu@polito.it

***Abstract*—The presence of foreign bodies in packaged food is a serious concern for both final consumers (allergies, injuries, choking) and food manufacturers (reputation and economic losses). In particular, low-density plastics, glass and wood splinters are hard to detect even by the most advanced X-ray imagers. One solution is Machine-Learning-based Microwave Sensing (MLMWS): a non-invasive, contactless, and real-time method which uses a machine-learning (ML) classifier to analyze the scattered microwaves from the irradiated target object. In this paper, we want to extend our previous work about contaminant detection in cocoa-hazelnut spread jars by proposing an enhanced ML flow to increase the accuracy of the ML classifier. For the first time in this case study, we use a multi-class classifier, we train it with scattering parameters measured at multiple microwave frequencies, with a new pre-processing scaler, data augmentation, quantization-aware training and a pruning schedule. The results show a contaminant detection multi-class accuracy of 94.167% with a latency of 26 µs when targeting an AMD/Xilinx Kria K26 FPGA. Finally, we released our datasets publicly to OpenML.[1]**

***Index Terms*—foreign body detection in food, machine learning, microwave sensing, neural networks, fpga acceleration**

I. Introduction

Foreign bodies in packaged food pose risks for consumers' health (e.g. injuries and choking) and could damage manufacturers' reputation and finances. Nowadays, there are many non-invasive techniques to detect foreign bodies in food (primarily metal detectors, X-ray, near-infrared, and terahertz imaging) [1], and each comes with some limitations [2]. In particular, low-density contaminants, such as wood, glass and some types of plastics, are very hard to detect for the current systems. One emerging solution, which has already demonstrated excellent detection performance [3–5], is Machine-Learning-based Microwave Sensing (MLMWS). This method takes advantage of the distinct differences in dielectric properties between food and contaminants. To perform the analysis, low-power electromagnetic waves at microwave (MW) frequencies are directed towards the object to analyze through a set of antennas placed at some distance from it. Then, the resulting scattered EM waves are recorded back by the same antennas and processed by a machine-learning (ML) classifier that checks for foreign objects. Since MLMWS is not an imaging technique, it does not require slow and computationally intensive algorithms to reconstruct the image of the inspected object. Thus, it is ideal for real-time and high-throughput food production lines [5].

The purpose of this paper is to expand upon our previous research on the detection of foreign bodies in hazelnut-cocoa spread jars using an MLMWS approach [4, 5]. In particular, using the datasets measured in [4], we propose an enhanced ML flow that improves classification accuracy. The novelties with respect to our previous works [4, 5] are the following:

1) We assessed the detection capabilities of a multi-class multi-layer perceptron (MLP) classifier, because it can help food manufacturers gather statistics on contaminant types that elude their industrial control systems. In this regard, we used MW frequency pairs as features, we augmented the training data by using Additive White Gaussian Noise (AWGN), and we analyzed the features of our datasets in detail to select a suitable feature pre-processing scaler.
2) We trained the best multi-class model using quantization-aware training (QAT) and compressed it with pruning to target an AMD/Xilinx Kria K26 FPGA with a multi-class accuracy of 94.167% and a latency of 26 µs.
3) We open-sourced our datasets of [4] to OpenML [6] to enable further ML research on this topic.

II. Related Work

Contrary to most prior related works, which use MW imaging to spot foreign bodies [2], in this work we use MLMWS. Recently, microwave-sensing (MWS) systems and ML tools have attracted the attention of the food industry. For example, SVMs for damaged apple sorting [7], MLPs for cherry defects [8] and, in particular, MLPs for intrusion detection in chocolate spread jars [4, 5], to which the present work is strictly related. In fact, it shares the same MWS system, dataset and model architecture (MLP) of [4], and some of the ML steps of [5], like *Bayesian Optimization* (BO) for hyperparameter search, quantization for model size reduction, and *hls4ml* [9, 10] for model deployment on an FPGA. However, the works presented in [4, 5] do not propose: 1) to train their classifiers with different MW frequency pairs as features ([4] used 10 GHz only, while [5] combined eleven frequencies from 9.0 to 11.0 GHz); 2) to use data augmentation; 3) to use *RobustScaler* for pre-processing data (instead of *StandardScaler* [4, 5]); 4) to train a multi-class MLP classifier (rather then a binary one [4, 5]); and 5) to train models with QAT and compress them with pruning (instead of post-training quantization only [4, 5]).

III. Microwave Sensing System and Datasets

In this work, we used the same MWS system of our previous research [4] as well as the five datasets collected at that time[2].

The MWS system consists of an array of six monopole antennas [11] that resonate at 10 GHz. The array is connected to a 2-port Vector-Network Analyzer (VNA) through a 2×6 custom-made electro-mechanical switching matrix [12]. The

[1]Fermilab report number: FERMILAB-PUB-23-313-PPD

[2]In [4], we collected five datasets, but opted to use only one.

979-8-3503-2711-3/23/$31.00 ©2023 IEEE

antennas are attached to an arch-shaped support lying above a typical industrial conveyor belt for packaged food. When the object to inspect is approaching the arch, a photocell triggers the acquisition process and the MWS system generates a 6×6 scattering matrix (*S-matrix*) of the object.

Each dataset was obtained by measuring 1200 contaminated and 1200 uncontaminated (i.e. *free jars*) samples. Each sample is an S-matrix of an irradiated hazelnut-cocoa spread jar, the target object of our work. The foreign bodies that we considered are both high-density and low-density: a metal sphere, a glass fragment, a big plastic sphere, a small plastic sphere, a triangular plastic fragment (*triangle*), and a cap-shaped plastic. The maximum dimensions of these range from 1 mm to 20 mm. Specifically, for each intrusion type, we measured 200 samples by varying the position of the contaminant in the jar. Moreover, all samples were acquired at five MW frequencies (9.0, 9.5, 10.0, 10.5, and 11.0 GHz), so that this process resulted in five S-matrices per jar and therefore five different datasets. After acquisition, each sample was converted to a feature vector to be used as input for the ML classifier. Each vector has 30 features corresponding to the sequence of the real and imaginary parts of the 15 upper diagonal scattering parameters of an acquired 6×6 S-matrix, excluding the monostatic elements in the main diagonal. We released our five datasets in the OpenML repository [6] with data IDs[3]: 455XX, XX = 36, 37, 38, 39, 40.

IV. Preliminary Machine-Learning Experiments

In this section, we elaborate on the need for an enhanced ML flow to improve our previous results on the same case study [4]. Also, we describe the preliminary experiments that helped us design the proposed flow that we present in Sec. V.

Initially, we analyzed the results of [4]. The binary MLP of this work was trained using the 10.0 GHz dataset and reached an accuracy of 93.958% on a balanced Test Set of 480 samples. However, a recall of 100% was achieved for all contaminants except for the plastic triangle, whose recall was 48.780% (20/41 samples). In fact, we defined free jars as the *negative class*, but contaminants as the *positive class*, so there were 21 *false negatives* (FNs). Nonetheless, this is unacceptable for an industrial scenario (ideally FNs = 0).

Consequently, we decided to assess a multi-class approach in this case study for the first time, because we believed that multiple outputs could help mitigate the error rate of the triangle. In addition, a multi-class output can help food manufacturers gather statistics on foreign bodies and identify the source of contamination in their production lines. Thus, we balanced the 10.0 GHz dataset for the multi-class scenario by keeping all 200 samples per contaminant and reducing the free jar samples to 200. Moreover, we used a splitting ratio of $80\%-20\%$ to develop and test with shuffling to create our sets [4], but we also stratified them based on the seven classes.

We then employed Scikit-Learn's `GridSearchCV()` and `StratifiedKFold()` to search for and evaluate multi-class MLPs using cross-validation. We optimized for hidden layers (from 1 to 3), neurons per layer (from 16 to 256), the optimizer (SGD, Adam or Adagrad), and the learning rate (from 0.001 to 100). We chose 200 epochs, a batch size of 32, and the activation function used by [4] (ReLU). As a result, we obtained a model consisting of a single hidden layer with 160 neurons, that used Adagrad with a learning rate of 0.3.

This model reached a multi-class accuracy of 88.571%, which equaled a binary accuracy of 98.571%, on our balanced Test Set of 280 samples (40 per class). Moreover, unlike [4], this MLP reached a recall of 95.000% (38/40 samples) for the triangle. However, despite these good results, after adding 200 unseen uncontaminated samples to balance our Test Set for the binary scenario, multi-class accuracy dropped to 67.083% (-24.261%) due to 129 *false positives* (FPs), where 124 of these free jar samples were mispredicted as triangle samples.

By analyzing these results, we realized that we needed to address three points: 1) overfitting; 2) the reduced size of our multi-class sets; and 3) our lack of insight into the statistics of our datasets. We began from point 2), specifically by adding back all the free jar samples to our 10.0 GHz dataset to maximize data usage. Then, using an $80\%-20\%$ ratio, we shuffled and split the entire dataset into a new Development Set (1920 samples) and a New Test Set (480 samples). Next, we shuffled and split the Development Set using a $75\%-25\%$ ratio into a New Training Set (1440 samples) and Validation Set (480 samples), to equal the size of the latter to the size of the New Test Set. Also, in every step, we stratified the sets.

To address point 1), we used `StratifiedKFold()` to refine hyperparameter ranges and analyze training and validation curves across folds to detect any split dependent behavior or overfitting. To explore a broad hyperparameter space, we used Scikit-Optimize's `BayesSearchCV()`, which implements BO with cross-validation. Moreover, we employed it to optimize for 16 hyperparameters, such as hidden layers (from 1 to 6), neurons per layer (from 4 to 2048) and layer dropout rates (up to 0.45). We repeated both methods various times as we slightly tuned the hyperparameters manually and evaluated the selected models post training to assess overfitting and validation metrics (by plotting training and validation curves). In our experiments, we also explored optimizers, learning rates, activation functions, epochs, batch sizes, MW frequency combinations (Sec. V-A) and data augmentation (Sec. V-B). Additionally, we experimented with Max-Norm regularization heuristics [13] and class weights, to address class imbalance after adding the free jar samples back as we previously described for point 2). Concerning point 3), we chose `RobustScaler()` after our data analysis (Sec. V-C).

V. Enhanced Machine-Learning Flow

Using the knowledge acquired through our previous experiments, we now propose an enhanced ML flow for MWS systems, which is shown in Fig. 1. Specifically, we applied it to the hazelnut-cocoa spread jar case study as a real example of food contaminant detection. Furthermore, based on the results in Sec. IV, we focused on a multi-class approach.

The steps or *blocks* reported in Fig. 1 appear in the same

[3]Visit: https://www.openml.org/d/<data_ID>.

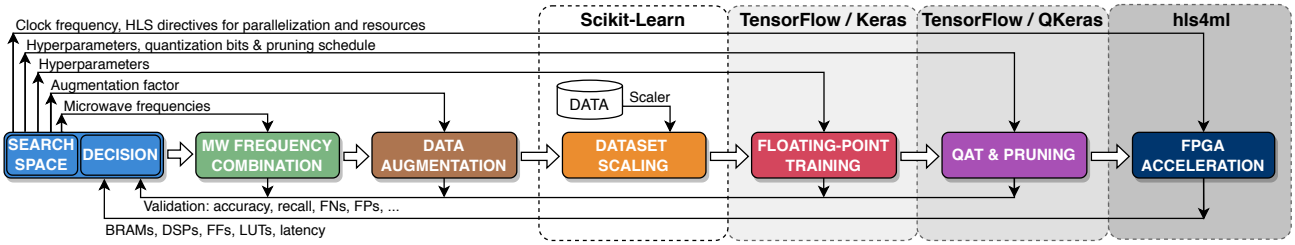

Fig. 1: The proposed enhanced ML flow for FPGA acceleration of neural networks.

order in which we suggest that they should be performed. However, for an MWS application, not all blocks may always be required. For instance, an ML designer might not perform augmentation if data is already sufficient for the application, or a hardware designer might not prune depending on the target FPGA. In particular, the first block, *Search Space and Decision*, is the core of the flow. This blue block: 1) samples inputs from the *search space* to send a selection to the other blocks; and 2) uses the metrics returned by the other blocks to *decide* on the next inputs. Its tasks can be manual or automated (e.g. by performing BO), depending on what aspect of the model needs evaluation (e.g. hyperparameter optimization). Now, we will explain the other blocks in more detail.

A) Microwave frequency combination. In the first step of our flow, we propose training the ML classifier with features derived from MW frequency combinations. Therefore, we explored different combinations of the five datasets described in Sec. III and compared our results to those achieved by training on these datasets separately. As an example, Table I summarizes the multi-class validation accuracy reached by our final multi-class Keras MLP, which we will present in Subsec.V-D, when trained with the scattering parameters of various MW frequencies. We observe that, by combining features related to different MW frequencies, we can achieve an accuracy increase of approximately +5% – +12%. We also observed accuracy improvements for other models that we considered. Moreover, similar results have been reported in [7], which describes a performance improvement of their MW imaging system when operating at three MW frequencies. Ultimately, based on our experiments, we chose the 9.5 and 10.5 GHz pair for the rest of our flow.

TABLE I: Multi-class results of some MW freq. combinations.

MW freq. (GHz)	Val. acc. (%)	MW freq. (GHz)	Val. acc. (%)
9.0	81.88	11.0	78.96
9.5	80.21	9.5 & 10.0	87.08
10.0	80.42	9.5 & 10.5	90.42
10.5	78.13		

B) Data augmentation. In the second step, we propose augmenting the Training Set to manage overfitting and improve the generalization of the model. In our case study, we performed augmentation by using AWGN to enlarge our 9.5 & 10.5 GHz Training Set of 1440 samples. Specifically, for each sample, we computed two normally distributed random variables for every $i\text{-}th$ scattering parameter s_i, one for the real part $\Re(s_i)$ and another one for the imaginary part $\Im(s_i)$, to add random Gaussian noise to every parameter with a Signal-to-Noise Ratio $SNR = 60\ dB$. We considered a mean $\mu = 0$, to not introduce bias, and a standard deviation $\sigma = \sqrt{\frac{S}{SNR}}$. Here, $S = (\sum_{i=1}^{N} \Re(s_i)^2 + \Im(s_i)^2)/N$, where $N = 30$ is the number of scattering parameters in a sample. Due to this augmentation step, an artificial sample is created from every original sample.

We explored the gradual augmentation of the entire Training Set to decrease FNs and FPs. During this process, to further decrease the FPs, we performed additional augmentation steps for the triangle class since the free jars were mostly confused with those containing the triangles. Ultimately, we chose an augmentation factor of 3 for all classes (i.e. 3×1440) and an additional augmentation factor of 18 for the triangles (i.e. 18×120). This was the best choice that reduced FPs and, at the same time, balanced the number of FNs and FPs in our experiments. Therefore, our New Training Set increased to 6480 samples. Among these, 2160 (33.33%) correspond to free jars, 2520 (38.89%) to triangles and 1800 to the other contaminants, where each one has 360 (5.56%) samples.

C) Dataset scaling. In the third step, we propose pre-processing data by using a scaler that addresses its statistics. We introduced this step because it should be present in every ML flow. However, this was not done in [4]. Therefore, we studied, for each single frequency dataset, the histograms of every feature. We found that most feature distributions did not resemble a Gaussian curve, possibly due to the reduced size of the datasets. We also obtained Box and Whisker Plots to visualize these distributions through quartiles, as well as the value ranges per feature. Although we found no outliers in terms of abnormal values, there were data points past the end of the whiskers. Moreover, while studying the feature value ranges, we observed that values differed by at least one order of magnitude. All the previous observations were also valid for the 9.5 and 10.5 GHz pair, before and after augmentation. Consequently, we chose to pre-process our data using Scikit-Learn's `RobustScaler()`, which uses statistics that are calculated per feature and are robust to data outliers. For every feature, this scaler removes the median of the feature and then divides the result by its Interquartile Range, which is the difference between the 75th and 25th percentiles.

D) Floating-point training. The fourth step comprises the same steps of the last paragraph of Sec. IV, in particular BO, stratified cross-validation and manual training. As a result, we obtained our final multi-class model, which is an MLP with four hidden layers with 300, 151, 195 and 128 neurons. Moreover, we used dropout rates of 0.4, 0.2, 0.1 and 0.05 and the

Scaled Exponential Linear Unit (SELU) activation function. We trained it considering 350 epochs, a batch of 320, the Adam optimizer and a learning rate of approximately 5.5e-5. We also employed `compute_class_weight()` by Scikit-Learn with the parameter `class_weight='balanced'` to account for class imbalance. Additionally, we performed regularization by leveraging the `MaxNorm` class by Keras to limit the weight norms to 4, as described in [13].

On our New Test Set of 480 samples (Sec.IV), our classifier reached a multi-class accuracy of 96.458% and a recall of 85.000% (34/40 samples) for the triangle class. Table II, instead, reports the most relevant binary-equivalent test metrics of our Keras model (row 2), from which we observe +3.334% in accuracy with respect to the binary MLP of [4] (row 1).

E) Quantization-aware training and pruning. In the fifth step, we suggest optimizing for hardware resources to improve FPGA deployment (Sec. V-F). Specifically, we considered QAT with QKeras [14] and pruning with a TensorFlow pruning schedule. QAT allows for training with reduced bit precision of weights, biases and activations, while pruning compresses MLPs by removing irrelevant neurons, so these methods speed up inference and lower power consumption. However, upon introducing these techniques, we had to check if the learning rate, epochs, batch size and activation function of our model would still manage overfitting properly. For this reason, we manually explored some values in [1.0e-5, 5.5e-5], [290, 780] and [290, 320] for the learning rate, epochs and batch size, respectively. We also considered decreasing the learning rate while increasing the epochs [15]. Regarding activations, we evaluated `quantized_relu()` and `quantized_tanh()` since SELU is not available in QKeras.

To quantize, we considered all the possible combinations of the following: 1) for activation layers, we varied the total number of bits from 8 to 18 using multiples of 2 and no integer bits; and 2) for weights and biases, we considered 8 bits with 2, 3 or 4 bits for the integer part and $alpha = 1$, which is a scaling factor defined in [14]. Thanks to quantization, we removed Max-Norm regularization. Regarding pruning, we chose `PolynomialDecay()` by TensorFlow to prune the best quantized model gradually during training. We applied an initial sparsity of 50% at step 2000 and a final sparsity of 75% around step 15120, where the steps are calculated by dividing the number of training samples by the batch size and then multiplying this result by the number of epochs.

Although our best model used `quantized_tanh()`, we had to replace this function with `quantized_relu()` and remove the fourth hidden layer to fit it on our target FPGA. Therefore, we trained it again using 700 epochs, a batch of 300, a learning rate of 5.5e-5, 16 bits for the activation layers (with no integer part) and 8 bits for weights and biases (with 3 integer bits). This quantized and pruned model achieved a multi-class accuracy of 94.167% on our New Test Set (Sec. IV) and a recall of 82.500% (33/40 samples) for the triangle class.

Table II also reports the binary equivalent results of our quantized and pruned model (row 3). As expected, accuracy decreased with respect to our floating-point model (Sec. V-D),

TABLE II: Binary vs multi-class classifiers (test metrics).

Best ML classifier	Binary Test. acc (%)	Binary FNs / FPs	Recall for triangles (%)
Binary ([4])	93.96	21 / 8	48.78
Multi-class (Sec. V-D)	97.29	6 / 7	85.00
Multi-class (Sec. V-E)	95.42	7 / 15	82.50

specifically by 1.875%. Regarding FNs, our models reduce the issue of the plastic triangle mentioned in [4] (as highlighted by column 3 and 4), at the cost of increasing FPs. Nonetheless, FPs are not as critical as FNs for this industrial food scenario.

F) FPGA acceleration. In the last step, we perform hardware acceleration to speed up the model inference. In fact, we considered our model from Sec.V-E to support the throughput of an industrial production line. We adopted hls4ml, an open-source Python framework for the co-design and translation of machine learning algorithms into hardware implementations [9, 10]. hls4ml translates the model into C++ specification for AMD/Xilinx Vivado HLS [16], which in turn generates a hardware implementation at the register-transfer level for the traditional synthesis and implementation flow targeting an FPGA as deployment hardware.

As a development board we chose the recently released AMD/Xilinx Kria KV260. This board is an ideal solution for edge deployment that combines programmable logic and an ARM multi-processor in the same system-on-chip (MPSoC). To fit the final model on the FPGA resources, we explored many hardware configuration, such as clock frequency and hardware parallelization. After deployment, the final model confirmed the multi-class test accuracy of 94.167%. For the final synthesis and implementation, we targeted a clock period of 5 ns that our model easily achieves on the Ultrascale+ FPGA fabric. Our accelerator has a streaming interface and a latency of 809 clock cycles (approximately 4 µs). When integrated with data movers to main memory and a software application to control the accelerator, the overall latency is 26 µs. The post-implementation resources are: 110,317 LUTs (94% of the total), 133,404 (57%) FFs, 656 (53%) DSPs, and 3 (2%) BRAMs. The estimated power consumption for the accelerator on the programmable logic is 0.86 W, while for the entire chip (including the SoC ARM cores) it is 3.27 W.

VI. Conclusion

In this paper, we proposed an improved machine-learning flow for microwave-sensing systems that, in the chocolate-spread jar case study, it helped increase the detection accuracy of foreign bodies compared to our previous work. In particular, the novelty that helped the most was the multi-class approach with a MW frequency combination and data augmentation.

VII. Acknowledgments

This document was prepared by Politecnico di Torino, Columbia University, and Fermilab partially using the resources of the Fermi National Accelerator Laboratory (Fermilab), a U.S. Department of Energy, Office of Science, HEP User Facility. Fermilab is managed by Fermi Research Alliance, LLC (FRA), acting under Contract No. DE-AC02-07CH11359.

REFERENCES

[1] M. T. Mohd Khairi, S. Ibrahim, M. A. Md Yunus, and M. Faramarzi, "Noninvasive techniques for detection of foreign bodies in food: A review," *Journal of Food Process Engineering*, vol. 41, no. 6, p. e12808, 2018. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/jfpe.12808

[2] 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.

[3] J. Nohlert, T. Rylander, and T. McKelvey, "Microwave measurement system for detection of dielectric objects in powders," *IEEE Transactions on Microwave Theory and Techniques*, vol. 64, no. 11, pp. 3851–3863, 2016.

[4] 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.

[5] M. Ricci, B. Štitić, L. Urbinati, G. Di Guglielmo, J. A. T. Vasquez, L. P. Carloni, F. Vipiana, and M. R. Casu, "Machine-learning-based microwave sensing: A case study for the food industry," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 11, no. 3, pp. 503–514, 2021.

[6] J. Vanschoren, J. N. van Rijn, B. Bischl, and L. Torgo, "Openml: networked science in machine learning," *SIGKDD Explorations*, vol. 15, no. 2, pp. 49–60, 2013. [Online]. Available: http://doi.acm.org/10.1145/2641190.264119

[7] 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.

[8] D. Guyer and X. Yang, "Use of genetic artificial neural networks and spectral imaging for defect detection on cherries," *Computers and Electronics in Agriculture*, vol. 29, no. 3, pp. 179–194, 2000. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168169900001460

[9] FastML Team, "fastmachinelearning/hls4ml," 2021. [Online]. Available: https://github.com/fastmachinelearning/hls4ml

[10] J. Duarte *et al.*, "Fast inference of deep neural networks in FPGAs for particle physics," *JINST*, vol. 13, no. 07, p. P07027, 2018.

[11] M. Ricci, F. Vipiana, and L. Crocco, "Microwave imaging system for in-line security assessment," in *2020 IEEE International Symposium on Antennas and Propagation and North American Radio Science Meeting*. IEEE, 2020, pp. 1139–1140.

[12] J. A. Tobon Vasquez, R. Scapaticci, G. Turvani, G. Bellizzi, N. Joachimowicz, B. Duchêne, E. Tedeschi, M. R. Casu, L. Crocco, and F. Vipiana, "Design and experimental assessment of a 2d microwave imaging system for brain stroke monitoring," *International Journal of Antennas and Propagation*, vol. 2019, pp. 1–12, 2019.

[13] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014. [Online]. Available: http://jmlr.org/papers/v15/srivastava14a.html

[14] C. N. Coelho, A. Kuusela, S. Li, H. Zhuang, T. Aarrestad, V. Loncar, J. Ngadiuba, M. Pierini, A. A. Pol, and S. Summers, "Ultra Low-latency, Low-area Inference Accelerators using Heterogeneous Deep Quantization with QKeras and hls4ml," Tech. Rep., Jun 2020, 9 pages, 9 figures, 3 tables, submitted to ICCAD. [Online]. Available: https://cds.cern.ch/record/2724942

[15] Qkeras/notebook at master · google/qkeras. [Online]. Available: https://github.com/google/qkeras/tree/master/notebook

[16] AMD/Xilinx, "Vivado HLS," https://www.xilinx.com/support/documentation-navigation/design-hubs/dh0012-vivado-high-level-synthesis-hub.html.