The Second Edition of European NO DIG Conference

Segrate (Milan), Italy
25TH May 2023

Paper 31

Trenchless Technologies aware Edge Computing for Pipes Leakage Detection

Fulvio Lo Valvo[1], Giacomo Baiamonte[1],
Prof. Ing. Giuseppe Costantino Giaconia[1]

[1] Dipartimento di Ingegneria, Università degli Studi di Palermo, viale delle Scienze – (edificio 9),
90128 Palermo, Italy
{fulvio.lovalvo, giacomo.baiamonte02,
costantino.giaconia}@unipa.it

Abstract. Correct resource management is nowadays a crucial issue, especially into the water management related field, due to its ever increasing importance. Smart embedded systems may greatly improve monitor capabilities especially in systems that are capable to host some Machine Learning (ML) algorithms. The hardware resources in that case are in terms of memory, footprint and time used to compute the ML tasks. These systems should be able to be both accurate and compact although the precision is directly proportional to the memory used for data storage. The focus of this work is the design of three ML models implemented in a microcontroller, with an application scenario devoted to monitor a Water Distribution Network by using vibrations input and trying to investigate their computational complexities and comparing the experimental performances obtained by each tested solution too.

1 Introduction

Nowadays almost half of the drinkable water is wasted while distributing it from the source to the user. The causes of leakages are several: pipes and infrastructures aging, the sign of wear, weather etc…

Due to the difficulties in recognizing the leak, and because the small leakages are the most common and the hardest to find, it takes a long time before the damaged pipe is detected and the problem solved. Acoustic instruments allow to find also these non-visible leaks, because the change in pipe's vibrations can be captured from an accelerometer sensor [1] and analysed by a ML hardware component.

The aim of this work is to compare the computational complexity and memory footprint on board of three ML models and evaluate the performance of these algorithms in the estimation of the leakage, when they are implemented in a constrained microcontroller-based environment.

The computation complexity is a measure of the number of compute resources, which are time and space, that the algorithm uses when running a compiled code. The time complexity measures how much time the algorithm takes to generate the output according to a predetermined input size. Space complexity, instead, measures the

quantity of memory the algorithm occupies to execute and store all the data according to a fixed input size.

Typical pattern recognition system includes features extraction. This step is determined during the development of the ML algorithm and is important to group the information into a single vector and increase the robustness of the system.

1.1 Edge Machine Learning

In recent years the diffusion of ML enabled computers to use models to find patterns and take decisions without requiring explicit programming. Due to the high requirements of its development, ML has been initially used in high power computing machines but the increasing interests in using its high versatility in distributed environments lead to industry investing in deploying ML in smaller, cheaper and low power devices.

The limits of these devices include low size memory and low computation capabilities, but their reduced size makes them the only option for devices that need to stay in small environments. The easiest way to deploy ML on microcontrollers is to develop regular ML models and then compress them into microcontroller libraries.

For this study, three ML models have been selected for investigation: the Multilayer Perceptron (MLP), the Support Vector Machine (SVM) and the Decision Tree (DT)

1.2 Multilayer Perceptron

An artificial neural network (ANN) mimics the functioning of biological neurons in the field of data processing [2]. MLP is frequently preferred for classification among the various types of ANN. The basic unit of the model is the neuron, which calculates the sum of weighted inputs and produces an output based on an activation threshold. The number of hidden layers between the input layer and the output layer varies depending on the depth of the network. Each hidden layer increases the network capabilities as well as its complexity.

1.3 Support Vector Machine

SVMs are suited both for regression and classification tasks, which is our case [3]. Unlike newer algorithms like neural networks, SVMs offer two key benefits: faster processing speed and superior performance with a limited amount of training data. This makes SVMs an excellent option for text classification problems where the number of training samples is low and not easy to expand.

1.4 Decision Tree

A rooted ordered binary tree (T) that is labelled with a variable x_i at each internal node and a value of either 0 or 1 at each leaf is known as a deterministic decision tree. An input x establishes the final leaf and thereby the output, in a deterministic manner. Computation begins at the root of the decision tree and ends at the final leaf. The complexity of this model is determined by the tree's depth, which represents the maximum number of queries made on the worst-case input. Further details on this type of ANN can be found in [4].

2 Hardware used

Audible noise produced by water leaks usually falls within the range of a few kHz. As this noise is in the form of a three-dimensional vibration, it can be detected by an accelerometer with a sampling rate higher than the bandwidth being sensed. The acquired signal can then be processed to generate a meaningful dataset, which can be further utilized by ML algorithms.

The dataset utilized in this study was gathered via laboratory experimentation, which involved a water circuit equipped with a pipe containing 4 taps of varying sizes and distances from the sensor, simulating different leak conditions [5]. An IIS3DWB accelerometer [6] was employed, which features an integrated digital conversion circuit capable of producing 26,6 ksps, with a configurable acceleration range set to ±2g. The evaluation board used in this study was equipped with an STM32L476RG microcontroller, which offers an ARM Cortex-M4 processor unit, 2MB of Flash memory, and 640KB of RAM.

These components are very small and have been chosen taking into account the possibility of inserting them into existing pipes during trenchless relining activities.

Sklearn [7] and keras [8] are the Python libraries that were utilized to develop and train machine learning models. The dataset was split into windows of 12288 samples each (equivalent to approximately 0,46 seconds of recording), resulting in a training set of 3600 signal windows, evenly distributed between leak and no-leak conditions. During training and validation, 3000 samples were utilized, while 600 were reserved for testing purposes.

The 3 features utilized in this study were automatically extracted. All these features have been computed starting from the acceleration modulus and gravity component removal. A scattering diagram showcasing these features is presented in Figure 1. The 3D diagram indicates that, despite having the same number of points for both classes, it is still possible to easily differentiate between leak and no-leak conditions.

The main tool used to evaluate the performance and memory footprint of each model on the microcontroller was STM32XCUBE-AI. To guarantee compatibility with the STM32 utility, models from sklearn were converted to ONNX (an open format

designed for representing machine learning models), whereas the MLP network created using the keras library was instead natively compatible with the tool.

To emphasize the distinctions between ML models executed on microcontrollers, the algorithms were not optimized for the application during training. This approach eliminates the impact of model optimization, and parameters such as computational complexity and hardware requirements were evaluated at each model type's worst case. For this reason, models were generated by exploring various activation functions and neural topologies until a suitable and equivalent classification error was achieved for each algorithm.

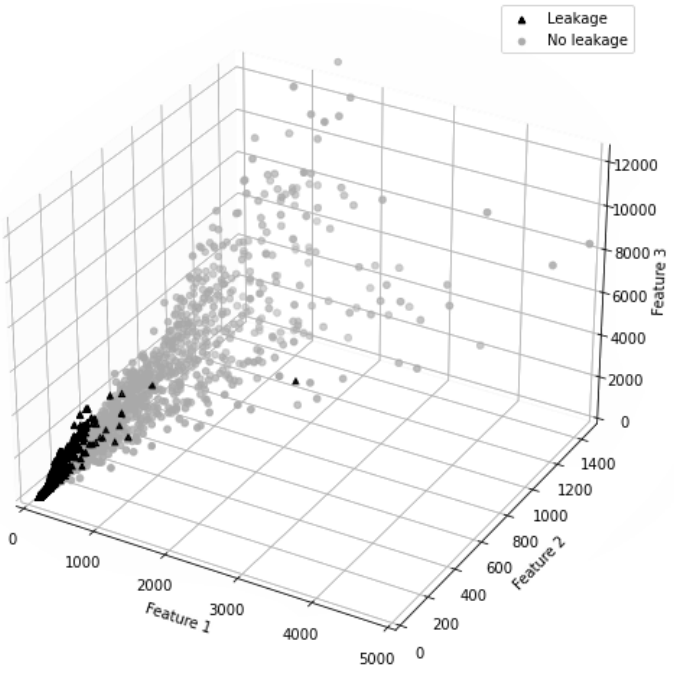

Figure 1 Diagram of the distribution of the extracted features.

The decision tree model was trained without any limitations on the possible number of branches and leaves. This can result in rules being generated for rare cases, which can lead to an increase in the size of the model. Consequently, the final trained decision tree model had 190 leaves and 379 nodes.

The SVM model required 767 iterations for training, and 584 leak and 604 no leak samples were used to calculate the SVM classifier.

The MLP network consists of two consecutive dense layers, each having 512 neurons, followed by a flatten layer. The selu (scaled exponential linear units) activation function was chosen for the dense layers, while the sigmoid activation function was chosen for the final layer.

This selection was based on the need for a small MLP network size and low inference error. Dropout layers were used during training to reduce overfitting.

3 Results

In order to demonstrate the differences in complexity between different models, they have been trained until the errors are comparable in at least one of the two classes. The characteristics of these models are presented in Table 1.

Table 1 Performance comparison among the three proposed models.

Model	Error on leaks	Error on no leaks	Number of MACCs	Ram (KB)	Memory flash (KB)
MLP	12%	8%	277515	6.09	1020
SVM	24.3%	5.3%	35640	8.13	48.16
DT	6%	8.3%	13	2.1	16.96

After training and converting the models with XCUBE-AI, their performance has been evaluated in a long-term scenario using a complete recording.

Figure 2 shows the outputs of the three models, where a value of 1 indicates a regular condition, while a value of 0 indicates leakage. The reference value reflects the true classification of the recording, where a tap was opened and then closed. The SVM model has the highest error rate in no-leak conditions, resulting in numerous false leak classifications, as shown in the corresponding plot.

The output signal of each of the ML algorithms is noisy, making it difficult to use in a real-case scenario. Therefore, it has been filtered by calculating the mean of 32 consecutive prediction values and assigning a value of 1 or 0 if the result is higher or lower than its middle value. Figure 3 shows the resulting filtered signals. The filter introduces a delay of approximately 15 seconds, which is relatively insignificant for the leakage detection use case, as a fast response is not required.

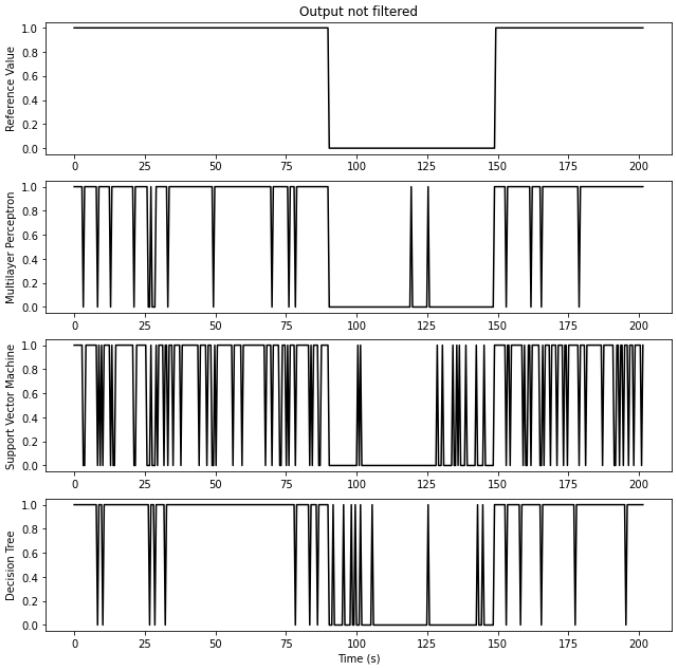

Figure 2 ML inference results related to the models.

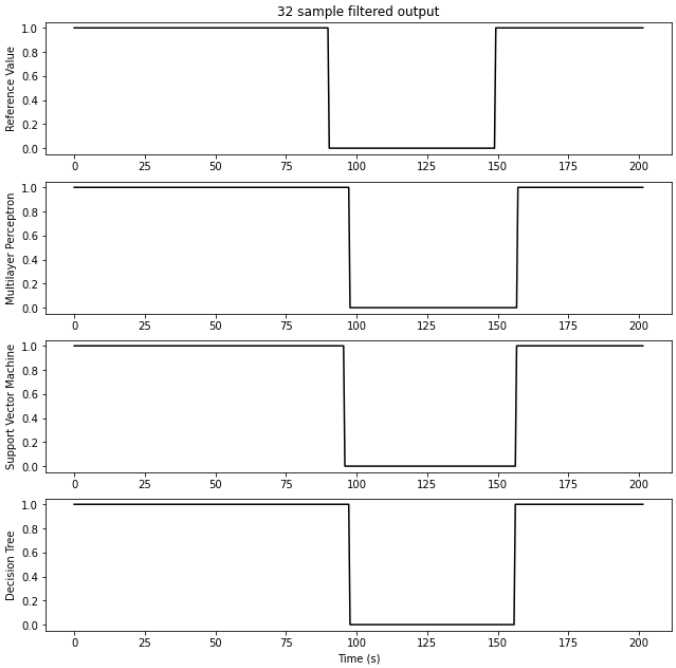

Figure 3 Filtered inference.

4 Conclusions

It is crucial to choose the appropriate ML algorithm for a given situation when attempting to face problems using ML techniques. In the context of embedded systems, this decision is also affected by memory usage, feature extraction, and energy limitations.

In the case of detecting leaks, the extracted features are highly indicative of the actual experimental data, which allows for the use of low-resource algorithms such as decision trees that can achieve comparable accuracy to more complex systems. To reduce noise in the output signal, the last step involves a filter that introduces a delay while increasing the algorithm's performance with a relatively low computational effort.

Future research will focus on analysing the computational complexity of the three algorithms in terms of execution time and CPU cycles, as well as exploring new ML models and feature extraction methods to enhance overall performance while reducing hardware resource requirements.

Acknowledgments This work has been partially financed by the Project "TiSento" (Azione 1.1.5. - POC Sicilia 2014/2020 Asse 1 - PO FESR 2014/2020).

References

1. L. Mistretta, G. C. Giaconia, A. Valenza, E. Napoli, C. Gianguzzi, M. L. Presti e F. d. Puma, «Embedding Monitoring Systems for Cured-In-Place Pipes,» Applepies, 2016.
2. Various, «Artificial Neural Network,» [Online]. Available: https://en.wikipedia.org/wiki/Artificial_neural_network.
3. B. Stecanella, «Support Vector Machines (SVM) Algorithm Explained,» [Online]. Available: https://monkeylearn.com/blog/introduction-to-support-vector-machines-svm/.
4. H. Buhrman and R. de Wolf, "Complexity measures and decision tree complexity: a survey," Elsevier, 2002.
5. F. Lo Valvo and G. C. Giaconia, "Algoritmi di Machine Learning implementati su Sistemi a Microcontrollore," Palermo, 2022.
6. STMicroelectronics, "Iis3dwb datasheet," 2020. [Online]. Available: https://www.st.com/resource/en/datasheet/iis3dwb.pdf.
7. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot and E. Duchesnay, "Scikit-learn: Machine Learning in Python," Journal of Machine Learning Research, 2011.
8. Chollet, Francois and a. others, "Keras," GitHub, 2015. [Online]. Available: https://github.com/fchollet/keras.