Received 4 April 2023, accepted 7 May 2023, date of publication 11 May 2023, date of current version 22 May 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3275447

RESEARCH ARTICLE

Online Black-Box Modeling for the IoT Digital Twins Through Machine Learning

RICCARDO CAROTENUTO[1], (Senior Member, IEEE), MASSIMO MERENDA[1,2], (Member, IEEE), FRANCESCO G. DELLA CORTE[3], (Senior Member, IEEE), AND DEMETRIO IERO[1,2]

[1]Dipartimento di Ingegneria dell'Informazione, delle Infrastrutture e dell'Energia Sostenibile, Università Mediterranea di Reggio Calabria, 89122 Reggio Calabria, Italy
[2]HWA S.R.L., Spin-Off Università Mediterranea di Reggio Calabria, 89122 Reggio Calabria, Italy
[3]Dipartimento di Ingegneria Elettrica e delle Tecnologie dell'Informazione, Università di Napoli Federico II, 80125 Naples, Italy

Corresponding author: Riccardo Carotenuto (r.carotenuto@unirc.it)

ABSTRACT Many applications involving physical systems, such as system control or fault detection, call for a behavioral, black-box, or digital twin of the real system. By observing input-output pairs, a nonlinear system's black-box twinning model can be built, thus enabling real-time accurate estimation of the system's health and status. We propose a modeling approach that can be implemented with little hardware resources and predicts system output with acceptable accuracy for a wide range of applications in the IoT and Industry 4.0 application domains, such as cloud and distributed predictive control, maintenance, fault detection, and model drift avoidance. This approach consists of building a compact numerical model, based on the concept of sum-decomposability, with reduced computational complexity and memory requirements, well suited for microcontroller-based IoT applications. The black-box modeling theory, the sizing process, and the learning method are reported. The outputs of two examples of non-linear systems are replicated in real-time using a pioneer experimental setup built around a microcontroller. According to experimental results, online learning and prediction are performed at 1 kS/s with a prediction error comparable to the resolution of the digitalized input-output data. The reduced size of the obtained model calls for real-time sharing and update with cloud and edge-based simulation ecosystems enabling a near real-time digital twinning of field systems.

INDEX TERMS Digital twin, microcontroller, non-linear dynamical systems, system predictor, black-box model.

I. INTRODUCTION

Advanced techniques such as predictive control and maintenance or fault detection require an accurate mathematical model of the system to be controlled, to synthesize the control action that achieves a certain objective or to compare the real performance with the nominal one, respectively () [1], [2], [3], [4]. However, an accurate analytical description of a given process cannot be obtained in many cases.

In some cases, however, a behavioral model is sufficient. The behavioral model does not require detailed knowledge of the internal mechanisms of the system, but is based on the observation of the input-output vector pairs and, knowing the sequence of the input samples, reproduces the temporal trend of the outputs [5], [6]. Behavioral models are often called black-boxes and can perform the functions of the digital twin (DT), i.e. to provide virtual replication of a physical system.

According to literature [7], the DT is a virtual representation or replica of an object, being, or system that can be continuously updated with data from its physical counterpart, thus synchronized with the field in a near to real-time fashion. A numerically-representable replica of a system in the form of a behavioral model that can be transferred, simulated and updated with real-time data represents *per se* a DT. This is the case of Battery Management Systems (BMS) for the automotive sector, motor and machinery predictive fault detection and maintenance [8], [9].

Digital twins of actual physical systems lay the way for cloud and distributed predictive control, maintenance, fault

The associate editor coordinating the review of this manuscript and approving it for publication was Giovanni Merlino.

detection, and model drift avoidance in the IoT and Industry 4.0 application domains.

In particular, some types of black-box models rely on some prior knowledge of the internal structure of the system they intend to replicate. They are tuned to the real system through the estimation of a set of parameters, while their internal structure is predefined [10].

In contrast, non-parametric black-box models require minimal prior knowledge of the real system, such as just the number of past time samples of the inputs and outputs required to reproduce the current output () [11], [12].

Black-box nonparametric models constructed from experimental input-output data sets may be easily incorporated by controllers in many industrial situations with very little engineering effort.

Much effort has been devoted to the construction of black-box models for the representation of dynamical systems using neural networks in many different fashions () [13], [14], [15], [16], [17], [18], [19]. However, the convergence of the training is not always guaranteed as well as the effective generalization capacity, which allows the model to provide a sufficiently correct output even for inputs not belonging to the training set.

Black-box models can be implemented by Machine Learning (ML) [20], [21], [22], [23], requiring reliable datasets, proper labeling, and a learning phase which is often quite costly in terms of time and energy.

Furthermore, all black-box models built with the different techniques are also vulnerable to drifting and aging, which makes it desirable to be able to "update" periodically and possibly automatically with current and online data from the real system.

Following a different approach, a black-box model can be seen as a static mapping function that is reconstructed by training using the learning data set. Unfortunately, this has so far implied exponential memory usage, for the mapping representation, in the number of inputs considered.

However, the recent availability on the market of powerful microcontrollers with adequate onboard memory [24] has finally made feasible the on-the-field implementation of the approach of the so-called sum-decomposability: the n-dimensional mapping, whose output depends on n inputs, is decomposed into the sum of a certain number of one-dimensional mappings, *i.e.* mappings that depend on a single input [25].

In this work, after a brief introduction to the sum-decomposable mapping approach, the first pioneering experimental implementation of the technique described in [23] is shown. A numerical model is sized and trained using an experimental setup that includes a predictor based on a NUCLEO-F767ZI board from STMicroelectronics, and two non-linear systems of example.

The paper is organized as follows. Section II describes the sum-decomposable mapping theory behind the black-box modeling; Section III introduces the representation of dynamical systems using static mappings; Section IV describes the experimental setup and tested circuits, whereas Section V reports the obtained results. Finally, Section VI draws the conclusions.

II. THE SUM-DECOMPOSABLE MAPPING THEORY: BRIEF NOTES

A multiple-input single-output (MISO) black-box model can be seen as composed of a pair: a static mapping function $f: \Re^n \rightarrow \Re$ from the input and the state space to the output space, and a suitable regression vector, which includes a certain number n of past input and output samples [26], [27]. Multiple Input Multiple Output (MIMO) black-box models can be obtained by juxtaposing Multiple Inputs Single Output (MISO) black-box models, one for each output.

When experimental samples are considered, they usually are the output of some kind of analog to digital converter with M quantization levels. In this way, the static mapping can be represented by an n-dimensional matrix, consisting of M^n discrete points. The direct representation of the n-dimensional point set is a very troublesome problem, because of the so called "curse of dimensionality", which takes place dealing with every nonparametric technique for mapping representation: in general, there is an exponential growth of the memory requirements in the number of the inputs. For example, with M = 256 (8-bit quantization) and $n = 10$, $M^n = 2^{80}$ memory locations should be allocated. Even for small M and n, this exponential growth makes this approach impracticable in a direct way in most cases. Moreover, the time required for the experimental acquisition of the input-output pairs also has an exponential growth.

A very large number of effective techniques aimed at reducing to a minimum the number of inputs have been proposed in past years (see for example [28], [29], [30]).

In this work, the approach developed in [25] to represent discrete multidimensional static mappings is followed. The "curse of dimensionality" is avoided by approximating the n-dimensional matrix with a proper set of 1-dimensional memory arrays. A machine learning algorithm computes the values contained in the arrays in order to minimize the mapping representation error.

More in detail, considering for the moment only single output systems, the mapping to be represented is:

$$y = f(x_1, x_2, x_n), \tag{1}$$

where $y, x_i \in \Re$, $i = 1, 2, \ldots n$.

In the following, and without complicating the formal notations used, we will consider all the variables as quantized with M levels.

In general, to obtain a sufficiently accurate mapping representation, an auxiliary variable vector $W = [w_1, \ldots w_N]$ generated by N ($N \geq n$) suitable linear transformations is required:

$$w_q = \boldsymbol{A}_q \boldsymbol{X} + b_q, \tag{2}$$

where $\boldsymbol{X} = [x_1, \ldots x_n]^{\mathrm{T}}$, $\boldsymbol{A}_q = [a_{q1}, \ldots a_{qn}] \in \Re^n$, $b_q \in \Re$, and $q = 1, 2, \ldots N$.

The approximation of f in suitable discrete points can be written as:

$$\tilde{f} = \sum_{q=1}^{N} \boldsymbol{g}_q \left(\boldsymbol{A}_q \boldsymbol{X} + b_q\right) = \sum_{q=1}^{N} \boldsymbol{g}_q \left(w_q\right) \approx f, \tag{3}$$

where $\boldsymbol{g}_q$ is a 1-dimensional array of M memory locations. Each array $\boldsymbol{g}_q$ is addressed by an integer index obtained after a proper quantization of w_q.

The higher the parameter N, the higher the accuracy of the representation of f given by (3). From (2) and (3) the computational complexity of the output generation is O(N).

The problem of choosing the optimal value for the parameter N is very similar to that of choosing the number of neurons of the MLP hidden layers. At present, only heuristic techniques are available; for example, N can be increased until an elbow in the decreasing representation error curve is reached, with a negligible improvement for larger values.

A noteworthy case is when a mapping f of n variables is decomposable. Mappings that are summation of non-linear functions G_q of a single variable x_q are naturally decomposable:

$$f = \sum_{q=1}^{n} G_q \left(x_q\right). \tag{4}$$

Excepting the quantization error, this kind of mapping is perfectly represented by the summation of n 1-dimensional arrays $\boldsymbol{g}_q$. In other words, each $\boldsymbol{A}_q$vector of (2) has only one 1 in position q, and zeros elsewhere, and $b_q = 0$:

$$\tilde{f} = \sum_{q=1}^{n} \boldsymbol{g}_q \left(x_q\right), \tag{5}$$

where remember that x_q is quantized with M levels.

In order to compute the values contained in the N arrays $\boldsymbol{g}_q$ from a sequence of experimental input-output pairs, i.e. the mapping samples, the following iterative learning rule presented is used:

$$\boldsymbol{g}_q^{i+1}(w_q) = \boldsymbol{g}_q^{i}(w_q) + \alpha\{y - [\boldsymbol{g}_1^{i}(w_1) + \boldsymbol{g}_2^{i}(w_2) + \ldots \boldsymbol{g}_N^{i}(w_N)\}, \tag{6}$$

where i is the iteration index, $q = 1, 2 \ldots N$, $2/N > \alpha > 0$, and g_q^i is the array computed at i^{th} iteration.

The representation technique is applied through the following steps:

a) N 1-dimensional arrays $\boldsymbol{g}_q$, of M elements each, are allocated. The initial value of all the arrays' memory locations is set to zero;

b) the N inputs are quantized with M levels (i.e. each input value becomes an integer number ranging from 0 to M-1), where $M = 2^{\text{Nbit}}$ and Nbit is the bit number used to quantize the inputs w_q. Each integer input value is used to address a different memory location in each $\boldsymbol{g}_q$;

c) the contents of the addressed N memory locations, one for each $\boldsymbol{g}_q$, are read and summed up; the result is the current approximation $y_{pr} = g_1^{\mathrm{i}}(w_1) + g_2^{\mathrm{i}}(w_2) + \ldots g_N^{\mathrm{i}}(w_N)$ of the mapping output y;

d) the representation error is computed as $e_{pr} = y$ - y_{pr}, where y is the true mapping output; $\alpha \cdot e_{pr}$ is then accumulated by the iteration (6) in all the memory locations involved in the current approximated output;

e) steps from b) to d) are repeated for all mapping input-output pairs.

The definitive advantage of this technique over other similar ones is that the convergence of the representation (3) to the mapping (1) through of iteration (6) has been demonstrated, provided that $0 < \alpha < 2/N$.

Decomposable mappings in particular can be represented with a significant reduction in memory requirements without compromising the integrity of the mapping information. In fact, we only need to compute and store n 1-dimensional arrays $\boldsymbol{g}_q$, with a maximum storage of $n \cdot M$ memory locations, as opposed to the M^n memory locations needed by direct mapping representations.

It is important to note that the availability of numerous subsequent occurrences of the same input-output pair is required for the correct construction of the arrays $\boldsymbol{g}_q$, and that the algorithm can receive the pairs in any order (for example, sequentially or randomly). In addition, note that the algorithm only constructs the $\boldsymbol{g}_q$ locally for those regions of the mapping where input-output data are available.

III. FROM CONTINUOUS TIME DYNAMICAL SYSTEMS TO BLACK-BOX MODELS

Each deterministic discrete time dynamical system may be thought of as a static mapping f from a collection of past and current inputs and outputs to future outputs. Without loss of generality, in what follows we will consider a Single Input Single Output (SISO) system:

$$y(k+1) = f[y(k), \ldots y(k-p+1), u(k) \ldots u(k-q+1)], \tag{7}$$

where (q, p) are the numbers of past samples input and outputs required to represent the dynamical system, respectively (see Figure 1). The concept could be extended to MISO systems straightforwardly.

The aim of this work is to show that it is possible to obtain a replica of the behavior of the system under examination by building a numerical model through the above-mentioned learning technique. The model thus obtained is placed side by side with the real system, is fed by the same input, and provides a good approximation of the real system output.

For example, in the event of a failure in the real system, a difference can be immediately observed between the real and simulated outputs provided by the model. This difference might be used for various purposes, including the identification of faults and anomalies.

For this purpose, first of all, it is necessary to identify which are the meaningful inputs and outputs of the dynamical system under examination. Next, the desired degree of accuracy

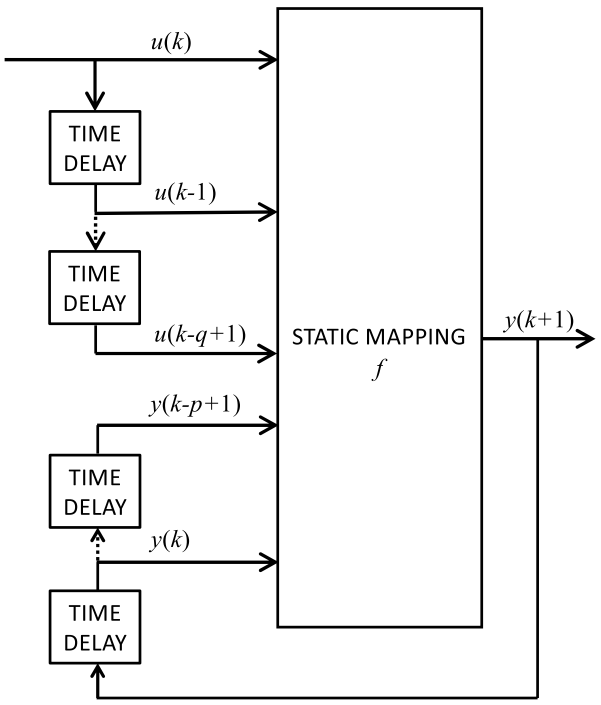


FIGURE 1. SISO discrete-time system represented as a static mapping and time delays.

in replicating the outputs is set by design. This in turn depends on two factors:

1) quantization level: a higher Nbit value decreases the quantization error. On the other hand, increasing *Nbit* increases the amount of memory and calculations required;

2) degree of decomposability in sum of 1-dimensional functions of the f to be reproduced: in case the f is not decomposable, adding array $\boldsymbol{g}_q$s improves the approximation, but, on the other hand, the memory usage and the required amount of calculations are increased.

A parameter to be chosen, which is connected to the number of levels M of each input, is the generalization parameter L. In fact, the greater the number of levels, i.e. array addresses, into which a given range of variation of an input is subdivided, the lower the probability that all available levels are used, i.e. all the memory locations of the $\boldsymbol{g}_q$ arrays are addressed, by the available input-output pairs. This can result in leaving some "gaps" in the contents of the arrays $\boldsymbol{g}_q$. The generalization mechanism "fills" these gaps using the "information" contained in a number of adjacent memory locations in the array.

The mechanism works as follows: instead of placing the value obtained from (6) only in the memory location addressed by the current input, a fraction $1/L$ of this value is stored in a certain number L of adjacent array locations. When generating the output corresponding to a certain input, the L memory locations adjacent to the corresponding index are added.

It has been shown that it is possible, after repeated iterations of this mechanism, to obtain a good approximation of the mapping not only for all inputs provided, but also for inputs for which the mapping has not been previously acquired [20], under the assumption that there is a certain "proximity" with the inputs actually supplied to the system. The greater the number of inputs provided and their density, and the smoother the mapping (i.e. continuous mapping and with continuous derivatives up to a certain order), the better the approximation that can be obtained.

Furthermore, the generalization allows to reduce the negative effect of mean zero noise on the input data, extending the mechanism of successive sums on a large base of arrays' memory locations. Since an analytical criterion for dimensioning the parameter L is not yet known, one proceeds by trial and error until a satisfactory result is obtained.

In order to build the SISO model the following steps are applied, according to Section II:

a) N 1-dimensional arrays $\boldsymbol{g}_q$, of M elements each (or $M+\ L$ elements in case of generalization L), are allocated. The initial value of all the arrays' memory locations is set to zero;

b) at sample times, past input and inputs are time shifted, i.e. $u(k) = u(k+1)$ and $y(k) = y(k+1)$;

c) new current system input $u(k+1)$ and output and $y(k+1)$ are sampled and analog-to-digital converted to integer numbers quantized with M levels (i.e. each value becomes an integer number ranging from 0 to M-1), where $M = 2^{Nbit}$ and *Nbit* is the bit number used to quantize both. Each integer input value is used to address a different memory location in each $\boldsymbol{g}_q$;

d) the predictor output is computed as $y_{pr}(k+1) = g_1[y_{pr}(k)] +, \ldots. g_p[y_{pr}(k-p+1)] + g_{p+1}[u(k)] + \ldots. g_{p+q}[u(k-q+1)]$;

e) the prediction error is computed as $e_{pr}(\text{k}+1) = y$ (k+1) - y_{pr}(k+1);

f) $\alpha \cdot e_{pr}$ is then accumulated by the iteration (6) in all the memory locations involved in the current approximated output;

g) steps from b) to g) are repeated (see flow chart in Figure 2, where learning is active when $\alpha > 0$).

When generalization is applied, i.e. $L > 0$, L read/write operations are executed on L adjacent memory locations starting from the current address, for each $\boldsymbol{g}_q$.

IV. EXPERIMENTAL SET-UP

The modeling technique presented here is applied for demonstration purposes to the construction of two black-box models of two distinct highly non-linear electronic circuits

A. EXAMPLE 1 - VOLTAGE CLIPPER CIRCUIT

The first circuit is a voltage clipper: the output is equal to the input only if it is higher than a certain voltage threshold, otherwise it has the same value as the voltage threshold.

In Figure 3 the circuit diagram of the circuit is shown, where the lower voltage threshold has been fixed at 1.5 V.

Figure 4 shows a 30 Hz sinusoidal 3 V$_{PP}$ and 1.5 V offset input and the circuit output, acquired via a digital oscilloscope (Keysight DSOX1204G).

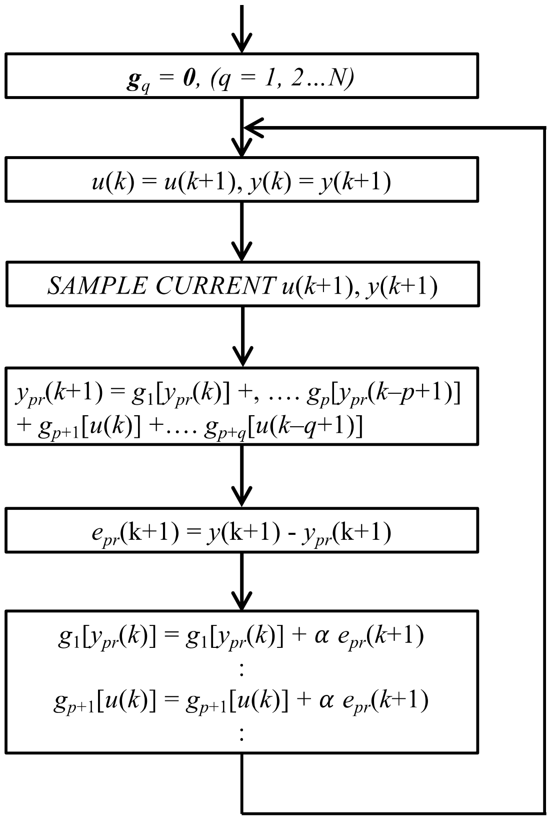

$g_q = 0, (q = 1, 2 \ldots N)$
$u(k) = u(k+1), y(k) = y(k+1)$
SAMPLE CURRENT $u(k+1), y(k+1)$
$y_{pr}(k+1) = g_1[y_{pr}(k)] +, \ldots. g_p[y_{pr}(k-p+1)] + g_{p+1}[u(k)] + \ldots. g_{p+q}[u(k-q+1)]$
$e_{pr}(k+1) = y(k+1) - y_{pr}(k+1)$
$g_1[y_{pr}(k)] = g_1[y_{pr}(k)] + \alpha\, e_{pr}(k+1)$
:
$g_{p+1}[u(k)] = g_{p+1}[u(k)] + \alpha\, e_{pr}(k+1)$
:

FIGURE 2. Flow-chart of the machine learning black-box algorithm. learning is active when $\alpha > 0$. When generalization is applied, for each g_q L read/write operations are executed on L adjacent memory locations starting from the arrays' current address.

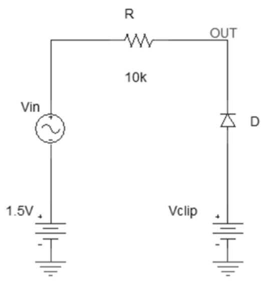


FIGURE 3. Voltage clipper circuit: the output voltage is limited down to Vclip= 1.5 V minus the diode forward voltage.

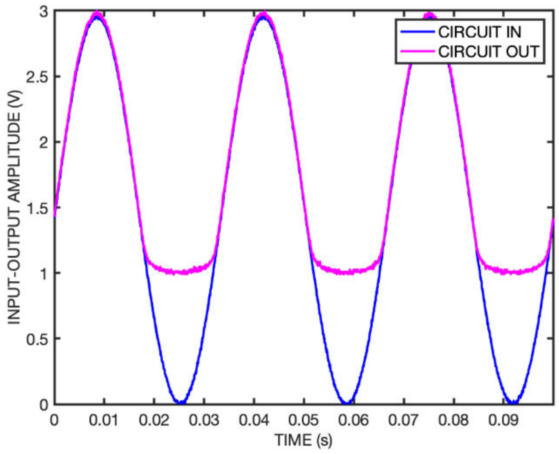


FIGURE 4. Experimental inputs and outputs of the voltage clipper circuit.

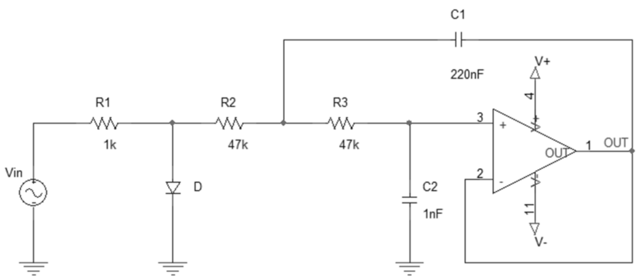


FIGURE 5. The nonlinear circuit: a diode limiter followed by a 2nd order low-pass filter with cut frequency 228 Hz and $Q = 7$.

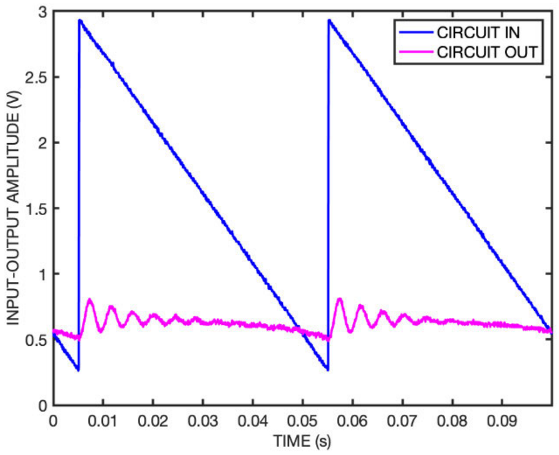


FIGURE 6. Experimental inputs and outputs of the 2nd order nonlinear circuit.

B. EXAMPLE 2 - NONLINEAR 2nd ORDER FILTER

Example 2 was designed to test the ability of the model built following the machine learning approach to reproduce the output of a dynamical system that exhibits high nonlinearity and damped oscillations.

The second circuit realizes a non-linear system consisting of a diode voltage limiter followed by an active 2ndorder low-pass filter with a cut frequency 228 Hz, and a Q value of 7, whose schematic is shown in Figure 5.

Figure 6 shows a sawtooth input and the circuit output, acquired by a digital oscilloscope. Figure 6 also shows that the system exhibits high nonlinearity and damped oscillations.

V. SIZING AND TRAINING THE BLACK-BOX MACHINE LEARNING MODEL: EXPERIMENTAL RESULTS AND DISCUSSION

In the following, two machine learning models of the two systems reported in the examples of the previous Section

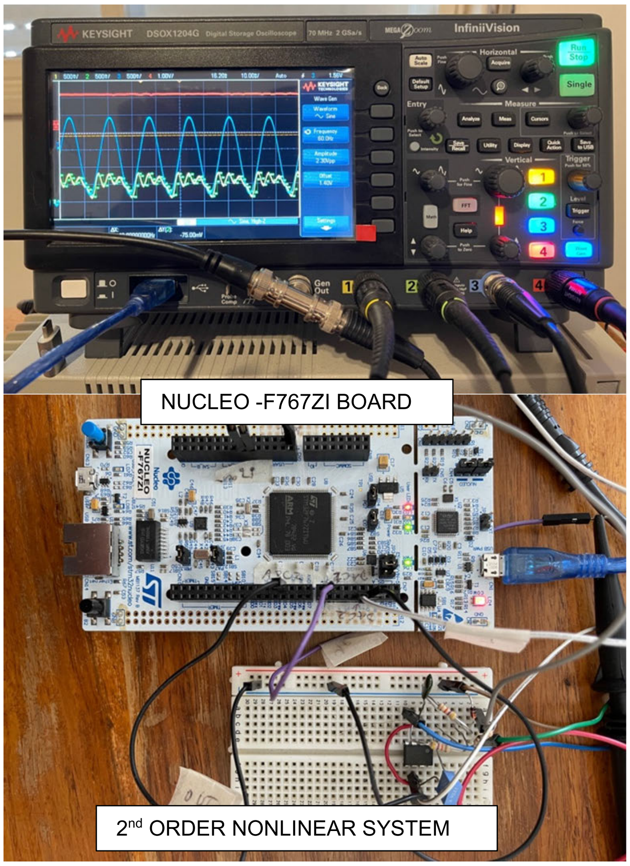


FIGURE 7. Experimental setup: up) Keysight DSOX1204G oscilloscope acquiring validation signals; down) the NUCLEO-F767ZI board acting as the predictor and the example non-linear electronic circuit system.

are built using experimental input-output data from the real systems.

First of all, the dimensioning activity of the black-box models is carried out, followed by the learning phase of the input-output relationship and by the validation that completes the process.

After the successful completion of the previous phases, for each of the two examples reported, a model is obtained whose outputs are a good approximation of those of the real system with the desired level of error.

Finally, in order to verify the ability of this approach to provide a way to detect a possible fault, a ‘‘fault’’ is induced in the first example real system, i.e. one parameter of the circuit is changed and the deviation between the outputs of the faulty circuit and of the model is evaluated.

The experimental black-box model is implemented using a NUCLEO-F767ZI board equipped with a STM32F767ZI microcontroller, with 32-bit ARM architecture, 512 kB SRAM, 2 MB Flash, and clock up to 216 MHz (see Figure 7).

In both circuits, inputs, and outputs are read by two analog-to-digital converters (ADC), integrated into the microcontroller, at a rate of 1 kS/s, while the resolution has been set to 8-bit. The output of the black-box model is produced at the same rate of 1 kS/s by one of the on-chip digital-to-analog converters (DAC), also with 8-bit signal resolution.

TABLE 1. STM32F767ZI usage data.

	value	unit
Supply voltage	3.27	V
Supply current	94.9	mA
Clock Frequency	216	MHz
RAM used	25.62/512 (5%)	kB
FLASH memory used	31.46/2048 (1%)	kB
ADCs' resolution used	8	bit
DACs' resolution used	8	bit
Sample time T_S	1	ms

The calculated sample-by-sample error is also output by the second on-chip the DAC, with a signal resolution of 8 bits.

In Table 1 is summarized the internal resource usage of STM32F767ZI (the RAM usage is related to Example 2).

A. EXAMPLE 1 - VOLTAGE CLIPPER CIRCUIT

Due to the instantaneous nature of the circuit of Example 1, it is assumed that the input-output relationship is the following:

$$y(k+1) = f[u(k)], \tag{8}$$

therefore, f is composed of one-dimensional function ($n = 1$).

Inputs and outputs samples vary in the range 0-3 V and are quantized with 256 levels (8-bit quantization).

The generalization parameter L was set equal to 128, i.e. half of the signals' range.

The microcontroller was programmed according to the steps depicted in Section III, where α was set at 0.07 (see Figure 2).

The real system and the MCU input are fed with the same voltage signal, while the MCU also samples the system output voltage.

When the model sizing is correct, i.e. the number of required inputs and outputs is correctly identified in relation to the structure of the system, together with the number of inputs and outputs passed, as the input-output pairs are presented to the predictor and processed according to the algorithm of Figure 2, the prediction error decreases gradually, up to the desired minimum value.

If, on the other hand, at the end of the training process the desired level of error is not reached, but it is believed that also the number of past instants used is correct or even over dimensioned, then most likely the system was not decomposable and one should resort to the more complex representation described in Section II by formulas (2) and (3), and follow the related sizing process.

The training phase can be considered completed only when the learning algorithm has received a sufficient number of input-output pairs whose values cover the entire range of

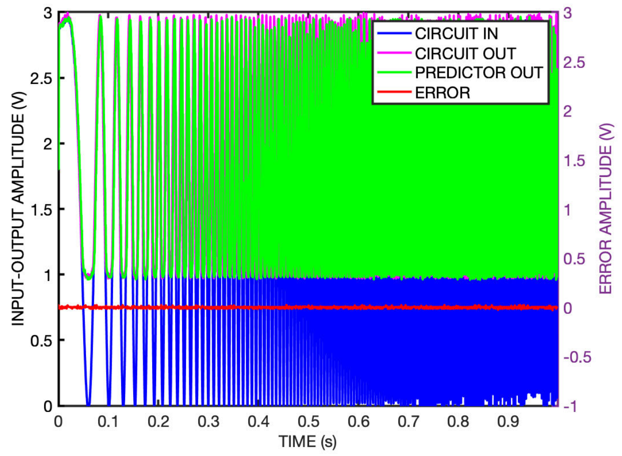


FIGURE 8. Learning signals: after the training phase, the model is able to reproduce the actual system output: input (blue), nonlinear circuit output (magenta), predicted output (green), and prediction error (red).

expected variation of inputs and outputs. By "expected variation range" we mean the system input-output signal range under the foreseen conditions of use.

Therefore, the duration of the training phase and the final level of the prediction error depend on the quantity, variety, or shape, of the input-output pairs that the learning algorithm receives as input.

The best input might seem to be the random type, which, as is known, contains within itself, if applied for sufficiently long times, signals of all possible shapes. However, in practice, this is almost never possible, both due to the long times necessary to observe some shapes of signal and because many real systems can undergo damage or dangerous behavior (e.g. robotic arms or navigation systems) if the input signal is not selected accurately.

In this example, the signal fed to the system during the learning phase, was a sinusoidal chirp up to 280 Hz, amplitude 3 $V_{PP,}$ and offset 1.5 V (see Figure 8).

Given the simplicity of the system considered, this signal certainly explores the whole range allowed for input and output.

In particular, Figure 8 reports the non-linear circuit input and output, the output predicted by our model, and the prediction error after the learning phase is completed. The prediction error is computed on-board the MCU on-line and it is the output of one of the two on-chip DACs. Since the DAC can't output negative voltage, the voltage corresponding to error zero is centered at about 1.6 V, i.e. the center of the output range of the DAC. For display purposes, the error is translated so that zero error corresponds to 0 V (see right y-axis).

Figure 9a shows that after 7 seconds (i.e. 7 kS) the percent learning RMS error went below 0.5% and after 10 s was below 0.25%. The limit RMS error was below 0.15%.

Figure 9b shows the validation phase, i.e. when the output of the predictor is based exclusively on the acquired

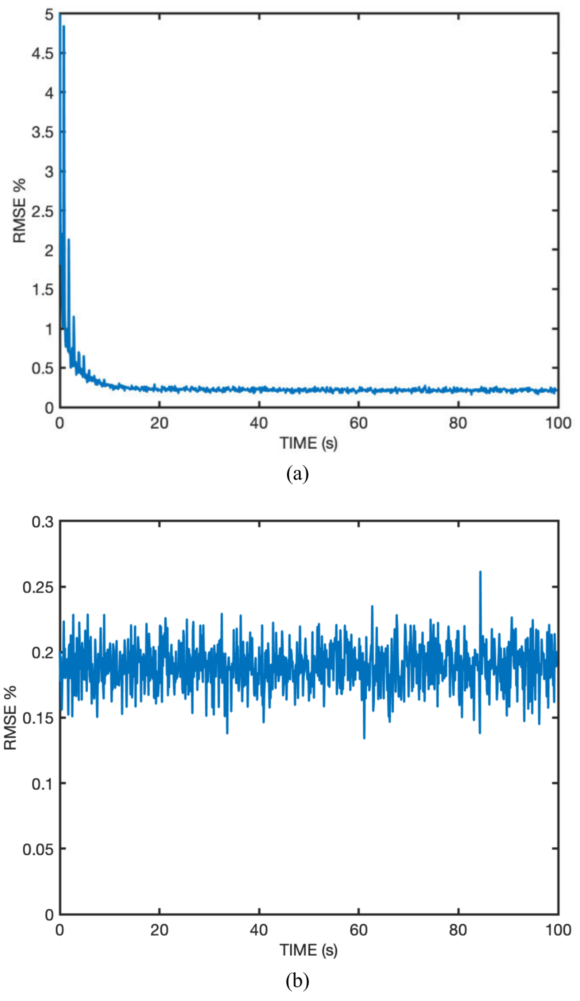


FIGURE 9. Experimental results: a) learning curve with RMS error decreasing along time; b) validation RMS error.

knowledge having set $\alpha = 0$. In this situation, the predictor receives only the input of the real system, and is able to replicate the output with a low error level comparable to that achieved in the training phase (see Fig. 9b). The final validation RMS error in all cases was below 0.3%.

Figure 10 shows the outputs in response to a signal of a shape not used in training. The prediction error is comparable with that obtained during the training phase.

Figure 11 instead shows the output signals when the original system is altered, for example due to a fault. Here the threshold voltage Vclip has been lowered from 1.5 V down to 1.15 V and the prediction error increases. The prediction error can be continuously monitored and when it exceeds a certain level can easily be used to detect a failure in the real system.

Figure 12 shows the content of the single $\boldsymbol{g}_1$ array at the end of the learning phase. This information is calculated onboard and stored in the RAM in real-time and is extracted through the debug port of the NUCLEO-F767ZI board using the STM32CubeIDE and STM32CubeMonitor programming tools for analysis and display purposes.

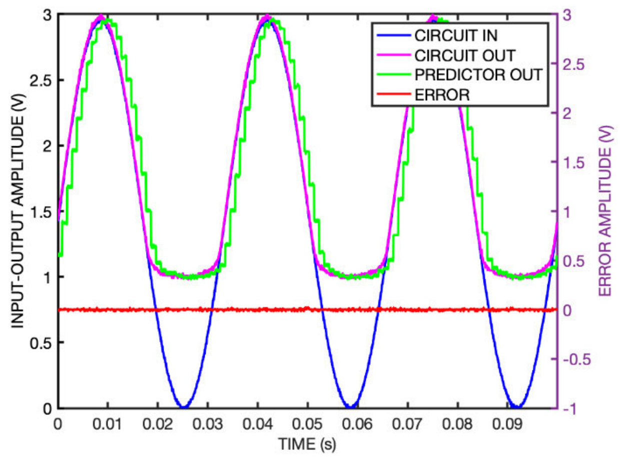


FIGURE 10. Outputs in response to a signal whose shape was not used during the training phase. The prediction error is comparable with that obtained during the training phase: non-linear circuit input (blue), circuit output (magenta), predicted output (green), and prediction error (red).

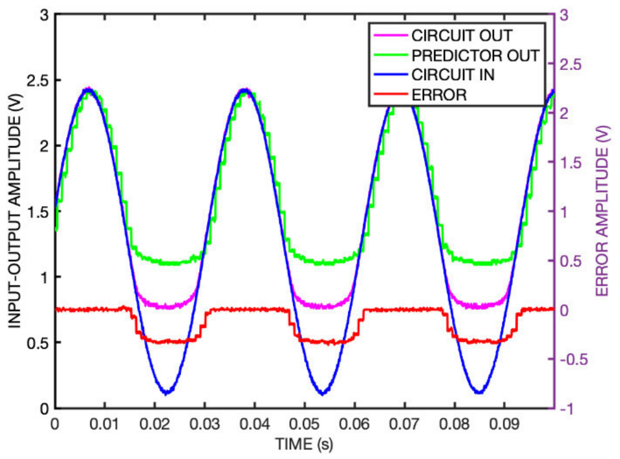


FIGURE 11. Experimental results after the original system is altered, for example due to a fault. Here the Vclip voltage has been lowered from 1.5 V down to 1.15 V, and the error becomes significative: non-linear circuit input (blue), circuit output (magenta), predicted output (green), and prediction error (red).

B. EXAMPLE 2 - NONLINEAR 2nd ORDER FILTER

The system under examination (see Figure 5) is composed of a non-linear section (the voltage limiter diode) and a linear section (the 2nd order filter). A careful analysis of the circuit shows that the nonlinearity acts only on the input, while the rest of the circuit is, within its saturation limits, linear.

Taking into account that we are dealing with a second order system, the model can therefore represent the system using a set of four arrays $\boldsymbol{g}_{1-4}$ indexed by the system's two past inputs and two past outputs:

$$y(k+1)=\boldsymbol{g}_1[y(k)]+\boldsymbol{g}_2[y(k-1)+\boldsymbol{g}_3[u(k)+\boldsymbol{g}_4[u(k-1)], \tag{9}$$

where arrays $\boldsymbol{g}_3$ and $\boldsymbol{g}_4$ are expected to represent nonlinear functions of $u(k)$ and $u(k\text{-}1)$, while the value of the contents of arrays $\boldsymbol{g}_1$ and $\boldsymbol{g}_2$ will represent linear relationships with their inputs, respectively.

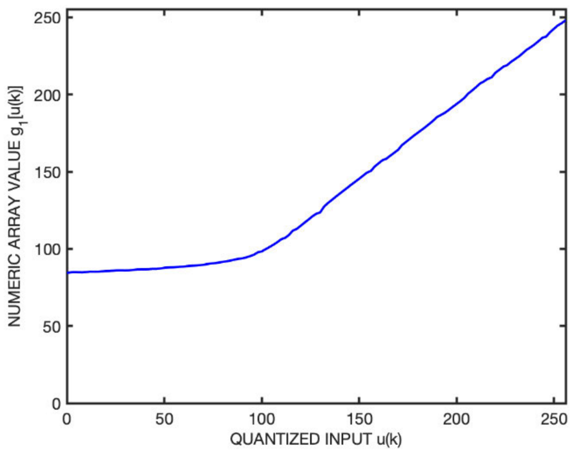


FIGURE 12. Content of the numeric array g_1 learned by the predictor that constitutes the numerical representation of the nonlinear system after the learning has been completed.

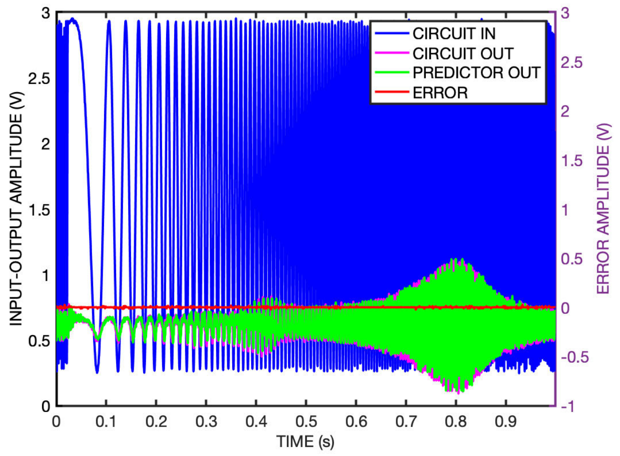


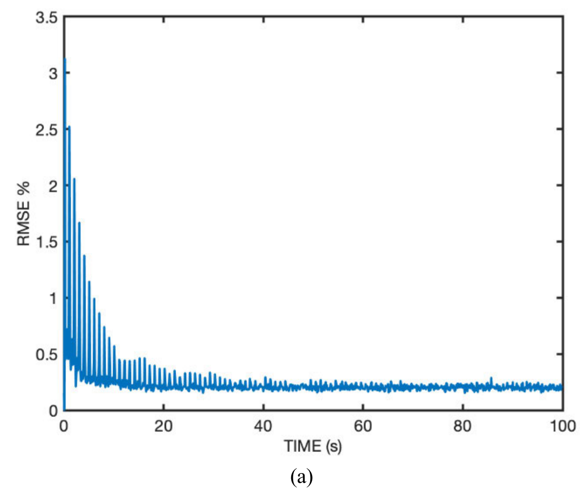
FIGURE 13. Learning signals: after the training phase, the model is able to reproduce the actual system output: input (blue), nonlinear circuit output (magenta), predicted output (green), and prediction error (red).

A signal, provided by the internal signal generator of the digital oscilloscope, is fed as input of the non-linear system.

In this case, the learning phase can also be carried out with simple input waveforms. A sinusoidal chirp up to 280 Hz, amplitude 2.65 V_{PP} and offset 1.575 V was used for the training (see Figure 13). The input signal was slightly reduced from the maximum amplitude (3 V) in order to avoid out-of-range outputs near the circuit resonance frequency.

Figure 14a shows that after 10 seconds (10 kS) the percent RMS learning error went below 1% and after 20 s was below 0.5%. The final RMS error was below 0.18%. Figure 14b shows the validation phase with $\alpha = 0$. The final validation RMS error in all cases was below 0.45%.

Figure 15 shows the predictor output after the learning and validation phases are completed and an input not used in the learning phase is used: non-linear circuit input (blue), circuit



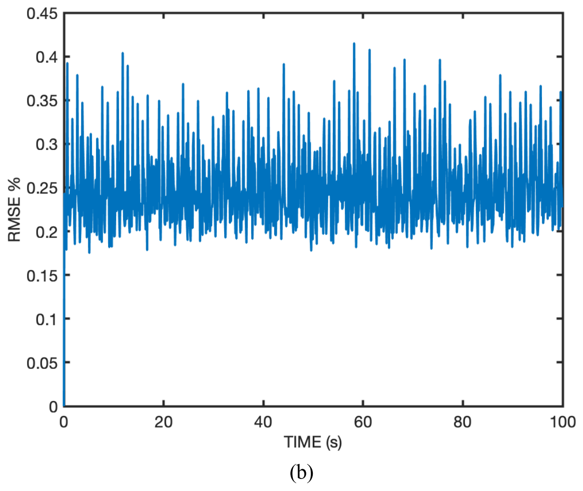
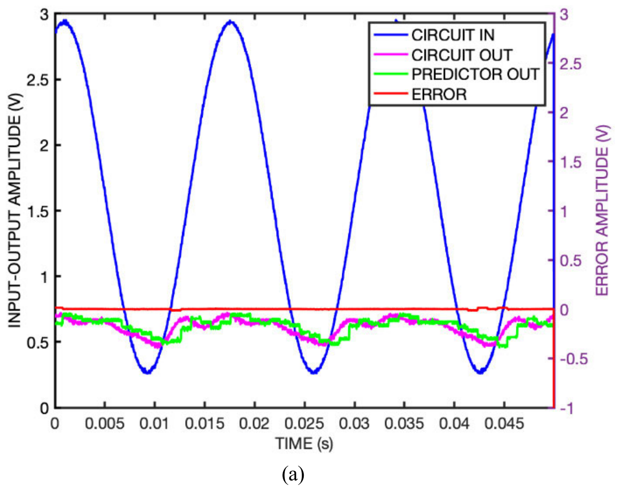


FIGURE 14. Experimental results: a) learning curve; b) validation error.



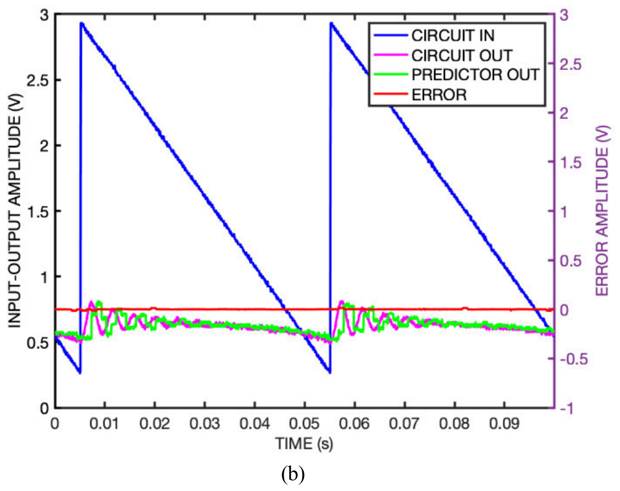


FIGURE 15. Validation: outputs in response to a signal whose shape was not used during the training phase. The prediction error is comparable with that obtained during the training phase: non-linear circuit input (blue), circuit output (magenta), predicted output (green), and prediction error (red) for two different circuit inputs: a) sine, and b) sawtooth.

output (magenta), predicted output (green), and prediction error (red) for two different circuit inputs: a) sine, and b) sawtooth. The prediction error is comparable with that obtained during the training phase.

Figure 14a shows that after 10 seconds (10 kS) the percent RMS learning error went below 1% and after 20 s was below 0.5%. The final RMS error was below 0.18%. Figure 14b shows the validation phase with $\alpha = 0$. The final validation RMS error in all cases was below 0.45 %.

Figure 15 shows the predictor output after the learning and validation phases are completed and an input not used in the learning phase is used: non-linear circuit input (blue), circuit output (magenta), predicted output (green), and prediction error (red) for two different circuit inputs: a) sine, and b) sawtooth. The prediction error is comparable with that obtained during the training phase.

On the basis of the good result obtained, for example if the error level reached is of the order of the LSB of the ADCs and DACs used, we can then, *a posteriori*, confirm that both the assumption of decomposability that we made in the sizing phase of the model, both the number of instants passed for the input and output samples are sufficient. As a thumb rule, a counter-test can be had by reducing the number of instants passed in order to verify if the final prediction error remains the same or increases. This heuristic procedure underlies the optimization process related to the size of the predictor, i.e. the number of $\boldsymbol{g}_q$s needed.

The fact that the relationship between inputs and outputs has actually been learned for all permitted inputs also has to be checked *a posteriori*.

At this point, one of the advantages that the proposed structure (6) has compared to other types of black-box models clearly emerges. In fact, the content of each array, which we can view at any time, represents the geometric projection of the n-dimensional mapping on the related independent variable. In the case of linear systems, trivially the trend of the values contained in each $\boldsymbol{g}_q$ is rectilinear with the variation of the addressing index, at least in the range of values received during learning. In the other cases, a more or less variable

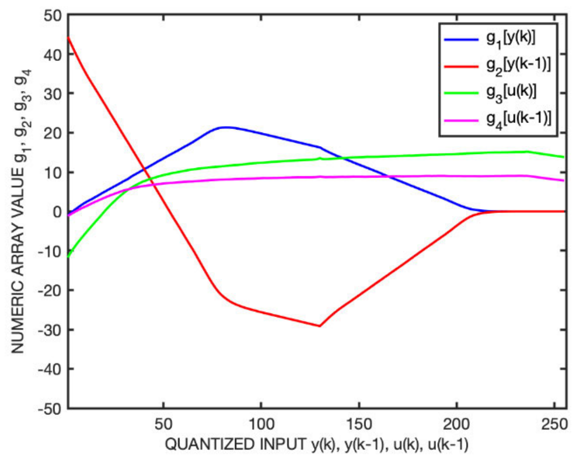


FIGURE 16. Content of the four numeric arrays learned by the model that constitute the numerical representation of the nonlinear system after the learning has been completed.

curve is obtained (see [25]). Typically, physical dynamical systems have rather smooth curves, sometimes interrupted by sharp discontinuities. With the proposed approach it is possible to see the portions of the arrays involved in the learning process and whether or not there are "gaps" or anomalies (saturations, etc.).

In fact, Figure 16 shows the content of four arrays $\boldsymbol{g}_{1-4}$ at the end of the learning phase. The information is calculated and stored onboard in real-time and extracted through the programming port of the NUCLEO-F767ZI board using the STM32CubeIDE and STM32CubeMonitor programming tools.

Arrays $\boldsymbol{g}_1$ and $\boldsymbol{g}_2$ during training have received inputs in the range of indexes 0-80 approximately corresponding to the output range of the system, which is limited in nature. Beyond the value of 80, the arrays have been filled by the generalization process (here $L = 128$) which obviously provides less and less accurate guesses moving away from the upper learning limit. Within the range of 0-80 they show a linear trend with respect to the index, as was expected.

Arrays $\boldsymbol{g}_3$ and $\boldsymbol{g}_4$, on the other hand, during training received inputs in the range of indexes 0-230 approximately corresponding to the range of the input fed to the system. Beyond the value of 230, the arrays have been filled by the generalization process, which obviously makes the guess less and less accurate as you move away from the upper limit of the learned region. Within the range of 0-230, they have a typical trend of a saturation curve such as the one expected from a diode limiter.

Once the arrays have been built correctly for the range being used, we can be sure that all inputs will produce correct outputs. It is therefore one somewhat "explicit" or "explainable" representation of the mapping. Such explainable interpretation of the numerical representation learned is clearly not obtainable using other representation methods such as MLP or similar, where the information on the input-output mapping is "implicitly" enclosed in the set of values assumed by the network weights and is not "visually" attributable to the modeled system properties.

VI. CONCLUSION

In this paper, a pioneer implementation of the black-box modeling of a nonlinear system based on the reduced dimensionality mapping theory using a resource-constrained microcontroller, was presented.

It was proved that the knowledge of relationship between a certain number of past inputs and outputs and the output to be predicted is sufficient for the sizing of the black-box model of the SISO nonlinear dynamical system.

A NUCLEO-F767ZI board was used in an experimental setting that showed online learning at 1 kS/s and operations with a low error, which is comparable to quantization error.

The learning is continuous or periodic, depending on operational needs.

The model obtained, a numerically-representable replica of the system, can be used, for example, for fault detection or predictive control and maintenance. The reduced size of the model allows for its real-time sharing and update with cloud and edge-based simulation ecosystems enabling a near real-time digital twinning of production systems while fed with current sensed data from the field.

The proposed technique has been showed well suited to being performed by microcontrollers with reduced computing capacity and relatively low power consumption, opening wide application in terrestrial, marine or aerial vehicles and IoT devices.

ACKNOWLEDGMENT

The authors gratefully acknowledge the kind support of STMicroelectronics, Catania, Italy.

REFERENCES

[1] Y. Pan and J. Wang, "Model predictive control of unknown nonlinear dynamical systems based on recurrent neural networks," *IEEE Trans. Ind. Electron.*, vol. 59, no. 8, pp. 3089–3101, Aug. 2012.

[2] J. Schoukens and L. Ljung, "Nonlinear system identification: A user-oriented road map," *IEEE Control Syst. Mag.*, vol. 39, no. 6, pp. 28–99, Dec. 2019.

[3] R. J. Patton, "Fault diagnosis in nonlinear dynamic systems via neural networks," in *Proc. Int. Conf. Control*, 1994, pp. 1346–1351.

[4] S. Pan and K. Duraisamy, "Long-time predictive modeling of nonlinear dynamical systems using neural networks," *Complexity*, vol. 2018, pp. 1–26, Dec. 2018.

[5] G. Pillonetto, F. Dinuzzo, T. Chen, G. De Nicolao, and L. Ljung, "Kernel methods in system identification, machine learning and function estimation: A survey," *Automatica*, vol. 50, no. 3, pp. 657–682, Mar. 2014.

[6] A. U. Levin and K. S. Narendra, "Control of nonlinear dynamical systems using neural networks. II. Observability, identification, and control," *IEEE Trans. Neural Netw.*, vol. 7, no. 1, pp. 30–42, Jan. 1996.

[7] M. Grieves and J. Vickers, "Digital Twin: Mitigating unpredictable, undesirable emergent behavior in complex systems," in *Transdisciplinary Perspectives on Complex Systems*. Cham, Switzerland: Springer, 2017, pp. 85–113.

[8] B. Wu, W. D. Widanage, S. Yang, and X. Liu, "Battery digital twins: Perspectives on the fusion of models, data and artificial intelligence for smart battery management systems," *Energy AI*, vol. 1, Aug. 2020, Art. no. 100016.

[9] G. Bhatti, H. Mohan, and R. R. Singh, "Towards the future of smart electric vehicles: Digital twin technology," *Renew. Sustain. Energy Rev.*, vol. 141, May 2021, Art. no. 110801.

[10] L. Ljung, *System Identification: Theory for the User*, 2nd ed. Upper Saddle River, NJ, USA: Prentice-Hall, 1999.

[11] L. Ljung, "Black-box models from input-output measurements," in *Proc. 18th IEEE Instrum. Meas. Technol. Conf., Rediscovering Meas. Age Inform.*, May 2001, pp. 138–146.

[12] L. Ljung, C. Andersson, K. Tiels, and T. B. Schon, "Deep learning and system identification," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 1175–1181, 2020.

[13] M. Norgaard, O. Ravn, N. L. Poulsen, and L. K. Hansen, *Neural Networks for Modelling and Control of Dynamic Systems*. London, U.K.: Springer-Verlag, 2001.

[14] S. Chen and S. A. Billings, "Neural networks for nonlinear dynamic system modelling and identification," *Int. J. Control*, vol. 56, no. 2, pp. 319–346, Aug. 1992.

[15] T. Chen and H. Chen, "Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems," *IEEE Trans. Neural Netw.*, vol. 6, no. 4, pp. 911–917, Jul. 1995.

[16] J.-Q. Huang and F. L. Lewis, "Neural-network predictive control for nonlinear dynamic systems with time-delay," *IEEE Trans. Neural Netw.*, vol. 14, no. 2, pp. 377–389, Mar. 2003.

[17] L. Lisang and P. Xiafu, "Discussion of stability on recurrent neural networks for nonlinear dynamic systems," in *Proc. 7th Int. Conf. Comput. Sci. Educ. (ICCSE)*, Jul. 2012, pp. 142–145.

[18] S. Mukhopadhyay and S. Banerjee, "Learning dynamical systems in noise using convolutional neural networks," *Chaos, Interdiscipl. J. Nonlinear Sci.*, vol. 30, no. 10, Oct. 2020, Art. no. 103125.

[19] P. Aivaliotis, K. Georgoulias, Z. Arkouli, and S. Makris, "Methodology for enabling digital twin using advanced physics-based modelling in predictive maintenance," *Proc. CIRP*, vol. 81, pp. 417–422, Jan. 2019.

[20] M. Merenda, C. Porcaro, and D. Iero, "Edge machine learning for AI-enabled IoT devices: A review," *Sensors*, vol. 20, no. 9, p. 2533, Apr. 2020.

[21] J. Pathak, Z. Lu, B. R. Hunt, M. Girvan, and E. Ott, "Using machine learning to replicate chaotic attractors and calculate Lyapunov exponents from data," *Chaos, Interdiscipl. J. Nonlinear Sci.*, vol. 27, no. 12, Dec. 2017, Art. no. 121102.

[22] Z. Qin, D. Sun, and C. Fan, "SABLAS: Learning safe control for black-box dynamical systems," *IEEE Robot. Autom. Lett.*, vol. 7, no. 2, pp. 1928–1935, Apr. 2022.

[23] S. L. Brunto and J. N. Kutz, *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*. Cambridge, U.K.: Cambridge Univ. Press, 2022.

[24] STMicroelectronics. *STM32F765xx STM32F767xx STM32F768Ax STM32F769xx Datasheet*. Accessed: Feb. 20, 2023. [Online]. Available: https://www.st.com/resource/en/datasheet/stm32f767zi.pdf

[25] R. Carotenuto, "A learning approximator for compact representation of experimental mappings," *Int. J. Adapt. Control Signal Process.*, vol. 17, no. 5, pp. 353–361, 2003.

[26] K. S. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE Trans. Neural Netw.*, vol. 1, no. 1, pp. 4–27, Mar. 1990.

[27] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1798–1828, Aug. 2013.

[28] E.-W. Bai, C. Cheng, and W.-X. Zhao, "Variable selection of high-dimensional non-parametric nonlinear systems by derivative averaging to avoid the curse of dimensionality," *Automatica*, vol. 101, pp. 138–149, Mar. 2019.

[29] S. Chen, S. A. Billings, and P. M. Grant, "Non-linear system identification using neural networks," *Int. J. Control*, vol. 51, no. 6, pp. 1191–1214, Jan. 1990.

[30] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, Dec. 2000.

RICCARDO CAROTENUTO (Senior Member, IEEE) was born in Rome, Italy. He received the Dr.Sc. degree in electronic engineering and the Ph.D. degree from Sapienza Università di Roma, Rome. He has been an Associate Professor of electronics with Università Mediterranea di Reggio Calabria, Reggio Calabria, Italy, since 2002. He has authored or coauthored more than 120 papers published in international journals and conferences proceedings and holds eight patents. His research interests include power conversion, energy harvesting, indoor localization, ultrasound imaging, ultrasound actuators, and neural networks theory and applications.

MASSIMO MERENDA (Member, IEEE) received the bachelor's, master's, and the Ph.D. degrees in electronic engineering from Università Mediterranea di Reggio Calabria (UNIRC), Reggio Calabria, Italy, in 2002, 2005, and 2009, respectively. From 2003 to 2005, he was a Fellow with the Institute of Microelectronics and Microsystems of the National Research Council IMM-CNR, Naples, Italy. From 2011 to 2018, he was a Postdoctoral Researcher with UNIRC. From 2021 to 2022, he was a Senior Scientist with the Cooperative Digital Technologies Competence Center, Austrian Institute of Technology (AIT), Vienna. From 2018 to 2021, he was a Researcher with UNIRC. He has been a Researcher with CNIT, since 2022. His research interests include the design of CMOS integrated circuits, RFID, silicon sensors, embedded systems, energy harvesting, the Internet of Things (IoT), and edge computing for the applications of the Internet of Conscious Things and beyond.

FRANCESCO G. DELLA CORTE (Senior Member, IEEE) was born in Naples, Italy. He received the M.S. degree in electronic engineering from Università di Napoli Federico II, Naples, in 1988. He was with Sirti spa, CNR-IMM, ENEA, and Optel-InP. He was a Full Professor of electronics with Università Mediterranea di Reggio Calabria, Reggio Calabria, Italy. He is currently a Full Professor of electronics with Università di Napoli Federico II. His research interests include integrated sensors, wide bandgap semiconductor device modeling, and silicon photonics. He has authored or coauthored more than 130 journal articles and holds three patents in his research area.

DEMETRIO IERO received the master's degree in electronic engineering from Università Mediterranea di Reggio Calabria, Reggio Calabria, Italy, in 2010, and the Ph.D. degree, in 2014. He is currently a temporary Researcher with the DIIES Department, Università Mediterranea di Reggio Calabria. His research interests include power electronics and switching power loss measurement, microcontrollers, embedded systems, the IoT, and RFID platforms.

Open Access funding provided by 'Università degli Studi di Napoli "Federico II"' within the CRUI CARE Agreement