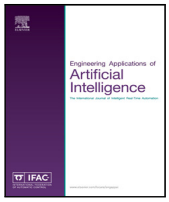
Engineering Applications of Artificial Intelligence 121 (2023) 105940

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai



# Deep neural networks for the efficient simulation of macro-scale hysteresis processes with generic excitation waveforms

Simone Quondam-Antonio [a], Francesco Riganti-Fulginei [b,*], Antonino Laudani [b], Gabriele-Maria Lozito [c], Riccardo Scorretti [d]

[a] Ferrari S.p.A, Maranello, Italy
[b] Department of Industrial Engineering, Electronics and Mechanics, Roma Tre University, Roma, Italy
[c] Department of Information Engineering, University of Firenze, Firenze, Italy
[d] Université Claude Bernard Lyon 1, Ampère-lab, Lyon, France

ARTICLE INFO

*Keywords:*
Deep neural network
Preisach modeling
Hysteresis simulation
Non-sinusoidal excitations

ABSTRACT

An effective and performing hysteresis model, based on a deep neural network, with the capability to reproduce the evolution of magnetization processes under arbitrary waveforms of excitation is here presented. The proposed model consists of a standalone multi-layer feed-forward neural network, with reserved input neurons for the past values of both the input (H) and output (M), aiming at the reproduction of the storage mechanism typical of hysteretic systems. The training set has been opportunely prepared starting from a set of simulations, performed by the Preisach hysteresis model. The optimized training procedure, based on multi-stage check of the model performance, will be comprehensively discussed. The comparative analysis between the neural network-based model, implemented at low level of abstraction, and the Preisach model covers additional hysteresis processes, different from those involved in the training. The mild/moderate memory requirement and the significant computational speed make the proposed approach suitable for a future coupling with finite-element analysis.

## 1. Introduction

The development and testing of performing hysteresis models, for the electromagnetic simulation of magnetic components and devices, are still an active and interesting field of research. Indeed, despite the large number of approaches already proposed in the literature, still nowadays none of them has been successfully and stably embedded in commercial computer-aided design programs. The material constitutive law is often approximated by means of bijective relations, such as the inverse tangent or the Langevin equation, neglecting the dependence on the past history (Viana et al., 2010). The lack of a general hysteresis model, which can be effectively coupled with Finite-Element Analysis (FEA) regardless the specific material and/or application, is due to strict requirements. According to FEA, the magnetic core is discretized in a 1-D or multi-dimensional mesh, and the hysteresis model is used to compute the constitutive law for each element. As a consequence, in addition to the accuracy, the reliability and the robustness, which are basic features, the models must be also efficient from the point of view of the computational time and the memory allocation. For each class of approaches already available, only a subset of the requirements listed above can be met successfully, and acceptable performances can be attained partially. For instance, the hysteron-based approaches turned out to be very accurate and robust, since they model the material using an arbitrarily large number of operators, suitably distributed and weighted (Lin et al., 2017; Leite et al., 2007; Cardelli et al., 2016a; Sarker et al., 2020). The Preisach Model (PM) is an example of well-established approach for the reproduction of hysteresis cycles in a wide excitation range (series of symmetric loops up to saturation) and with generic types of waveforms (in presence of sub-loops or DC bias), exploiting a constant hysteron distribution (Antonio et al., 2021b; Sarker et al., 2020; Salvini et al., 2003). Unfortunately, the large number of operators, required to produce smooth hysteresis loops in the full excitation range, reflects in a relevant increase of both the computational time and the memory allocation, which may be unacceptable for FEA. Some formulations have been proposed for the hysteron-based approaches to reduce the number of operators. An effective solution is offered by the Energy-Based Model (EBM) (Antonio et al., 2021b; Longhitano et al., 2019; Steentjes et al., 2017; Jacques et al., 2018), which is based on the Play-type hysteron. Unfortunately, despite the model identification leads to a minimal number of Play operators, a non-negligible loss of accuracy occurs in the reproduction of minor loops (Antonio et al., 2021b). An improvement of the EBM, proposed in Upadhaya et al. (2020), consists in the introduction of a

* Corresponding author.
*E-mail address:* riganti@uniroma3.it (F. Riganti-Fulginei).

https://doi.org/10.1016/j.engappai.2023.105940
Received 26 March 2022; Received in revised form 6 October 2022; Accepted 30 January 2023
Available online 9 February 2023

phenomenological coefficient to account for the inter-domain coupling. It allows a better reproduction of minor loops, but the calculation of the model output, that requires an iterative procedure, is more complex and less efficient from the computational view point. Globally, that formulation is less suitable for coupling with Finite Element equations. The Jiles–Atherton (JA) model (Jiles and Atherton, 1986) belongs to the class of approaches based on analytic equations, and it is perhaps the most famous of them. The original JA is based on a differential equation that has to be solved numerically, with five parameters to be identified from experimental data. The model is relatively simple and cheap from the point of view of the computational resources to be allocated, however some drawbacks make it not convenient for direct coupling with finite element analysis. Indeed, the original JA presents a high numerical accommodation, which is not found in many soft magnetic alloys for engineering applications. The numerical accommodation leads to a loss of accuracy in computing hysteresis cycles with minor loops (Leite et al., 2009; Laudani et al., 2014; Hussain and Lowther, 2017). In addition, it turns out that the reproduction of a family of symmetric hysteresis loops with different amplitudes cannot be effectively achieved by means of constant parameters (Fulginei and Salvini, 2005). Great improvements have been obtained by successive extension of the original JA, mostly exploiting magnetization-dependent parameters (Fulginei et al., 2015), increasing the complexity in coupling the model equation in FEA. More recently, novel approaches have been developed to reproduce the hysteresis loop branches via either polynomial interpolation techniques (Ram and Kulkarni, 2019) or rational and power functions (Mirzaei et al., 2021). These analytical methods are computationally efficient tools that turned out to be successfully applicable for different soft magnetic materials. Although they are relatively easy to identify from experimental data, they can only deal with purely symmetric hysteresis cycles and it is not easy to extend their formulation in case of non-sinusoidal and/or non symmetric excitations. The neural network-based models constitute another class of phenomenological approaches for computational magnetics (Adly and Abd-El-Hafiz, 2014; Chen et al., 2021; Makaveev et al., 2001; Laudani et al., 2015; Adly and Abd-El-Hafiz, 1998; Cirrincione et al., 2002; Sixdenier et al., 2008; Kuczmann and Ivanyi, 2002; Cardelli et al., 2016b; Antonio et al., 2021a) that attracts a lot of interest for their intrinsic high calculation speed coupled with poor memory request. The main issue in modeling hysteretic systems via neural networks (NNs) is the way to account for the past history. According to the theory of dynamic systems, a hysteresis model can be described in terms of two relations: the input-state equation, which implements the dependence on the past history, and the state-output equation, which is a memory-less function. The recurrent NNs have intrinsically the memory dependence, but they are difficultly adequate in modeling magnetic hysteresis, for the complexity of the training procedure (Adly and Abd-El-Hafiz, 2014; Chen et al., 2021; Makaveev et al., 2001). On the other hand, most of the NN-based models exploit the feed-forward architectures (FFNNs), which are well-established, extensively studied and some standard training algorithm are fully developed and available (Laudani et al., 2015). Several techniques have been developed to include the memory-storage mechanism. On one hand hybrid models have been proposed, in which the neural network, used to compute the memoryless state-output equation, is coupled with a hysteron-based model, which provides the memory dependent input-state equation (Adly and Abd-El-Hafiz, 1998; Cirrincione et al., 2002). In practice, the role played by the NN is alternative to the role played by the weight function, hence, the resulting model has more or less the same computational efficiency and requires the same amount of memory than the hysteron based model alone. For this reason, other approaches, seldom available in the literature, avoid the coupling of the NN with hysteresis models and exploit suitable algorithm to account for the past history dependence (Makaveev et al., 2001; Sixdenier et al., 2008; Kuczmann and Ivanyi, 2002). The model proposed in Kuczmann and Ivanyi (2002) consists of three FFNNs, a matrix containing some information related to the turning points of the first-order reversal curves and an algorithm that manages the selection of the neural network at each simulation step. More recently, we have proposed an algorithm based on the "transplantation method", that allows the NN model to reproduce stable minor loops (Antonio et al., 2021a). Again, the dependence on the past history is artificially modeled by an algorithm. In order to give a contribution in this field of research, we have developed a hysteresis model based on a standalone neural network, capable to solve arbitrary sequences of applied magnetic fields without any additional algorithms or computational strategies to account for the memory dependence. The neural network is a multi-layer (deep) FFNN, with reserved input neurons for the past values of both the magnetic field (input) and the magnetization (output). In this way the dependence on the past history is directly embedded in the neural network, whose architecture is again of feed-forward type, thus allowing the use of traditional learning algorithms. It has been found out that the generation of the training set by a hysteresis model, suitably identified from a series of measured loops, is more convenient than directly training the FFNN on the experimental data. In particular, exploiting the PM, few quasi-static symmetric hysteresis loops, measured under sinusoidal magnetic inductions, are sufficient to train the FFNN successfully (Antonio et al., 2021a). The main scope of this work is to present a hysteresis model based on a standalone deep FFNN with intrinsic memory dependence, capable to reproduce the PM simulations under various types of excitations, seeking more at a general methodology rather than a specific application. For this reason, the parameters of the PM are not identified for a given material, they are instead chosen to produce generic smooth hysteresis loops, with coercivity and remanence similar to those of the Fe-based soft ferromagnetic alloys. The Preisach model is here adopted to generate both a suitable training set and other data set to be used for the validation of the NN-based model, including symmetric hysteresis cycles, saturation cycle with sub-loops and DC biased magnetization loops. The soft ferromagnetic materials are indeed subjected to this types of excitations in several equipment and devices for power electronics (Corti et al., 2019; Locorotondo et al., 2019; Antonio, 2019; Coco et al., 2013). A detailed description of the PM formulation and an overview on the computations performed are shown in Section 2. To model the memory-dependence, a suitable architecture of the FFNN has to be identified, in particular: the number of hidden layers and the number of neuron-per-layer, the types of inputs and output. The architecture has been determined experimentally, checking the evolution of the training error and defining the under- and over-training limits. The past history is here taken into account by some input neurons which are dedicated to the current and the previous values of both the input magnetic field H and the output magnetization M. The training set adopted here is a saturation (major) loop with equally-spaced sub (minor) loops distributed along its branches. Each minor loop individuates a sector of first-order reversal curve (FORC) and a sector of second-order reversal curve (SORC). Furthermore, the inversion points that belong to the major loop branches can be reached in two different ways: either along the branch of the major loop or along the closing branch (SORC) of an inner loop. Training the neural network on this process allows to distinguish the future value of M dependently on the past values of both H and M, i.e. to model the dependence on the past history. Section 3 reports the formulation of the NN-based model and the optimized technique developed to train it. After that the FFNN has been successfully trained, the simulation of other magnetization loops, different from those previously obtained by the PM for the training, is made with the aim to verify the capability to generalize the patterns applied and to reproduce generic hysteresis processes. The robustness and the computational stability of the model is also checked by the simulation of hysteresis loops obtained undersampling the sequence of the applied magnetic field. The comparison of the results, presented and discussed in Section 5, also covers the computational cost of the two models, both implemented in Matlab® computing environment at low level of abstraction.

**Table 1**
Hysteresis properties assumed: coercive field, remanence, magnetization values corresponding to $H = 1000$ A/m and $H = 2000$ A/m , respectively.

| $H_c$ [A/m] | $\mu_0 M_r$ [T] | $\mu_0 M_{1000}$ [T] | $\mu_0 M_{2000}$ [T] |
|---|---|---|---|
| 30 | 0.75 | 1.50 | 1.59 |

**Table 2**
Parameters of the weight function identified by the GA.

| Parameter | $\sigma_{H1}$ | $\sigma_{H2}$ | $\sigma_U$ | $\mu_0$ | $\alpha$ |
|---|---|---|---|---|---|
| Value | 15 A/m | 500 A/m | 45 A/m | 22 A/m | 0.7 A/m |

## 2. Preisach model simulations

The classic Preisach model has been utilized to generate a series of hysteresis processes for the training and validation of the proposed NN-based model. The data sets obtained by PM simulations represent the reference magnetization loops we are interested to reproduce and they can be effectively considered as virtual experiments. This approach was preferred not only because the simulated processes are easier to obtain and cleaner respect to those directly measured on a device, but also because the PM can be identified successfully from a minimal set of experimental loops and can be eventually involved as intermediated stage for the FFNN training from measured data (Antonio et al., 2021a). Here, the characteristic features of the PM are set to reproduce a virtual material having the magnetic properties reported in Table 1.

At first, the PM has to be discretized by means of a suitable numerical distribution of Preisach operators on the magnetic field axis. Here, the two degrees of freedom chosen to represent univocally each operator are the "interaction field" Hi, i.e. the magnetic field at which it is centered, and its intrinsic coercivity u. The points of the grid relative to the interaction field follow the rule described by the equation below:

$$\begin{cases} H_i\left(n+\dfrac{N_H}{2}\right) = \left[ n\left(\dfrac{2}{N_H}\right) H_{max}^{\frac{1}{\beta_H}} \right]^{\beta_H} \quad n = 1,2,\dots \dfrac{N_H}{2} \\ H_i(n) = -H_i\left(N_H - n + 1\right) \end{cases} \tag{1}$$

where $H_{max} = 10^4$ A/m, $N_H = 2000, \beta_H = 1.7$. According to the power law (1) the points are dense nearby the origin, where most of the Barkhausen jumps occur, and become sparser for increasing values of the interaction field. Similarly, the progression of the coercive field values is

$$u(n) = \left[ \frac{n}{N_U} u_{max}^{1/\beta_U} \right]^{\beta_U} \quad n = 1,2,\dots,N_U \tag{2}$$

where $u_{max} = 500$ A/m, $N_U = 500, \beta_U = 2$. The maximum number of operators is $N_H \times N_U = 106$, whilst the saturation field (i.e. the field at which the last Barkhausen jump occurs) is $H_{SAT} = H_{max} + u_{max} = 10\,500$ A/m. After that the grid $(H_i, u)$ is defined, the value of the weight associated to each point of the grid $P(H_i, u)$ has to be determined. Here, the weight function, called also distribution function, is approximated with Lorentzian probability distributions, applying the principle of variable separation, as follows:

$$P(H_i, u) = \left[\alpha L_{H1}(H_i) + (1-\alpha) L_{H2}(H_i)\right] L_U(u) \tag{3}$$

where $\alpha$ is the parameter of the linear combination, to be identified, while the functions $L_{H1}$, $L_{H2}$ and $L_U$ are the 1-D Lorentzian functions listed below.

$$L_{H1,2}(H_i) = \frac{1}{\pi} \cdot \frac{\sigma_{H1,2}}{(H_i^2 + \sigma_{H1,2}^2)}, L_U(u) = \frac{1}{\pi} \cdot \frac{\sigma_U}{(u-u_0)^2 + \sigma_U^2} \tag{4}$$

The dependence of $P(H_i, u)$ on the magnetic interaction field is reproduced by a linear combination of two Lorentzian functions centered at the origin of Hi axis, having different parameters, in order to have more degrees of freedom in defining the hysteron distribution with respect to Hi. On the other hand, a single Lorentzian function with offset u0 is applied to represent the hysteron distribution with respect to the intrinsic coercivity. The quantity u0 is the most probable value of the hysteron intrinsic coercive field. The parameters $\sigma_{H1}, \sigma_{H2}, \sigma_U, u_0$ and $\alpha$, have been determined to comply with the properties shown in Table 1 using the genetic algorithm (GA), described in Antonio (2019), which returns the values listed in Table 2. The properties shown in Table 1 are not for a specific material but they are assumed by the authors. The numerical magnetization obtained as the weighted sum of the contributions of all the hysterons is scaled such that the value of $\mu_0 M = 1.59$ T for $H = 2000$ A/m.

The identified weight function is plotted in Fig. 1. One can note that the maximum value of P is found for $(H_i = 0, u = u_0)$. The simulation of all the magnetization loops always starts from the non-magnetized state: the first magnetization curve is traced from 0 to the maximum magnetic field in a number of samples $N_{fmc} = 25$, after that two periods are simulated in order to complete the loop and let its first and last points coincide. A detailed description of the hysteresis processes simulated by the Preisach model is reported in the following.

- DATASET 1: major loop with extreme values $\pm H_{extr} = 2000$ A/m, having 36 inner loops equally distributed along its ascending and descending branches. The inversion points, in A/m, on the descending branch of the major loop, from Hextr to $-H_{extr}$ are described by $IP^{des}$ = [525,187,53,16,4.5,−4,−11.5,−17.5,−24, −31.5,−40,−49,−60,−77,−104,−158,−268,−602]. The inversion points along the ascending branch are $IP^{asc}(i) = -IP^{des}(i)$ for $i = 1,2,\dots,18$. The sequence of the applied field H(k) is linear between each couple of local maximum and minimum points and the number of samples per period is SPP = 5652.
- DATASET 2: major loop with $H_{extr} = 2000$ A/m and 12 inner loops equally distributed along its ascending and descending branches. The inversion points do not coincide with the ones of the previous data-set, indeed $IP^{des}$ = [6,−6,−20,−22,−27,−200]. Again, the behavior of $H(k)$ is linear between each couple of maximum and minimum points, while SPP = 4040.
- DATASET 3: family of 7 symmetric hysteresis loops with the following extreme values of the magnetic field $H_{extr}$ = [20,35,57, 100,220,562,1000]. The behavior of $H(k)$ is triangular and SPP = 1000.
- DATASET 4: family of DC-biased magnetization loops. The magnetic field sequence consists of a constant bias value superimposed to a sinusoidal component having different amplitudes:

$$H(k) = H_{DC} + H_{AC} \cdot cos\left(2\pi\left[\frac{k-1}{SSP-1}\right]\right) \tag{5}$$

  with SPP = 400, $H_{DC}$ = 800 A/m and $H_{AC}$ = 200,300,400,500, 600,700 A/m.
- DATASET 5: the sequence has a sinusoidal shape with exponentially-decreasing amplitude, starting from $H_{extr} = 1000$ A/m. The mathematical expression of the sequence is reported below.

$$H(k) = H_{extr} \cdot e^{\left[\frac{-2\pi}{\tau} \cdot \frac{k-1}{SPP-1}\right]} \cdot cos\left(2\pi \frac{k-1}{SSP-1}\right) \tag{6}$$

  with $k = 1,2,\dots,n_p \cdot SPP$ and $\tau = 4\pi$.

The number of periods traced is $n_P = 10$, while SPP = 600. By observing the structure of the Preisach model and its equations, you easily understand what are the critical points to consider for a practical application. Indeed, to obtain good results in terms of simulation, this model needs of a quite large number of hysterons. This means a high memory footprint that must contain all the states (frozen state or not frozen state) characterizing the distribution of hysterons. Moreover, because the algorithm must consider all hysterons everytime it calculates a single value of magnetization, the computational costs are very high (for more details regarding this aspect, please see Scorretti et al., 2022).

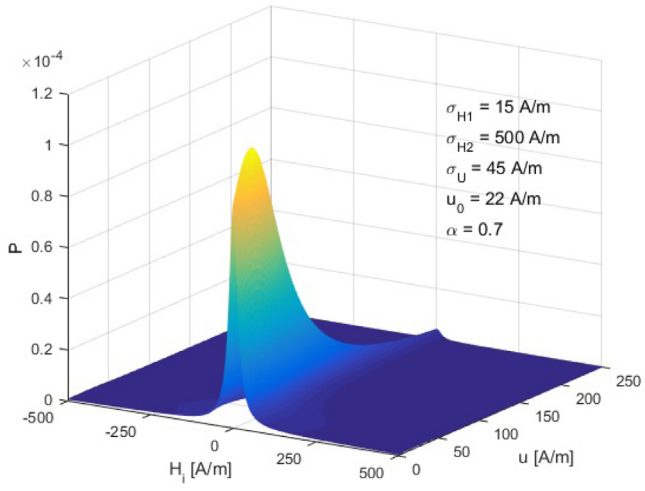



**Fig. 1.** 2-D plot of the weight function.

## 3. Neural network model: Definition and training

The NN-based model has been developed with the aim to include the past history dependence directly on the network, which should be able to reproduce arbitrary hysteresis processes without any additional algorithm or computational strategy.

### 3.1. Model definition

The architecture that we considered is a deep feedforward with 4 hidden layers, each one having 12 artificial neurons. The memory dependence is modeled by reserving four inputs to represent the current (at sample step k) and the past (at sample step k − 1) magnetization states of the system, which are given in terms of the coordinates on the H-M plane. The model has totally five inputs: H(k), M(k), H(k − 1), M(k − 1), plus the next value of the magnetic field applied H(k + 1). The output is the current susceptibility $\chi(k)$, from which the future value of the magnetization can be easily obtained: $M(k+1) = \chi(k) \cdot [H(k+1) - H(k)] + M(k)$. Fig. 2 shows the functional block diagram of the NN-based model. As one can note, a FIFO buffer is used to store the current value (Mcur) and the previous value (Mold) of the magnetization, which must be sent back as inputs for the next iteration. When the future value M(k + 1) is calculated, Mcur shifts on the left and becomes the old value, while M(k + 1) is stored as Mcur. The "closed loop" configuration we have described is the typical operation mode during the simulations.

In order to exploit in a convenient way the capabilities of the proposed architecture, a suitable training procedure must be carried out, starting from the definition of the training set. For example, a family of FORCs, that turned out to be effective to train conventional FFNNs, is not appropriate in our case. Indeed, the FORCs do not carry any information about the possibility to reach a given point of the H-M plane from different directions. The inversion points (IPs) on the main loop branch, from which the FORCs originate, are not relevant because the two different magnetization values that can be reached are characterized by opposite sign of the increment dH. It has been seen that, when the sign of dH changes, the future M is calculated exploiting the symmetry on the M-H plane rather than directly by the FFNN (Kuczmann and Ivanyi, 2002; Antonio et al., 2021a; Coco et al., 2013). A more suitable dataset for training must contain few points reached by different direction without any change of the sign of dH. The principle of memory dependence on which the NN-model is based, is represented in Fig. 3.

In addition, each curve of the training set must not give redundant information with any of the others, and the total information supplied should be sufficient to let the FFNN reproduce conventional hysteresis processes, such as the symmetric cycles and the first magnetization curve. The solution proposed here is to use a major hysteresis cycle with a series of inner loops. In this way, each inversion point on the main cycle branch can be reached from either along the main cycle branch itself or along the sector of SORC that is the closure branch of the inner loop. In both the cases, the magnetic field does not change direction. Among the PM simulations, only the DATASET 1 has been involved in the training of the FFNN, while all the others have been used to validate the model. The magnetization curves relative to the DATASET 1 cannot be immediately used. First of all, it is convenient to normalize both the magnetic field and the magnetization to 1. The normalized quantities are indicated with lowercase letters. Furthermore, thanks to the antisymmetric character of the susceptibility, only the part of the training curves with the same sign of dH can be considered. In facts, if the magnetic field changes direction, the value of the susceptibility can be calculated as $|\chi(H, M, dH)| = |\chi(-H, -M, -dH)|$. For this reason, the final sequences h(k) and m(k), applied for the training, have been obtained from the normalized data by selecting only the descending branches of the curves, reducing to 2826 the number of points, as illustrated in panel (a) of Fig. 4. The sequence $\chi(k)$, displayed in the panel (b) of the same Figure, is calculated as $[m(k+1) - m(k)]/[h(k+1) - h(k)]$ for each $k = 1, 2, \ldots, 2825$.

### 3.2. Optimized training procedure

The input sequences $h_{cur}(k), m_{cur}(k), h_{old}(k), m_{old}(k)$ have been defined from the training set sequences h(k), m(k) already described, selecting the samples either from 3 to 2826 (current fields values) or from 2 to 2825 (old field values). On the other hand, the output sequence $\chi(k)$, to represent the current value of the susceptibility at each sample k, has to be truncated from 2 to 2825. The conventional methods for the training of feedforward networks consist of running traditional algorithms for a given number of epochs, appropriately dimensioned on the basis of the net size and architecture (number of hidden layers and neurons). The evaluation of the training performance is based typically on the behavior of the error function, such as the mean absolute error (MAE) or the mean squared error (MSE), with respect to the epochs. However, the performance of the trained FFNN cannot be described only in terms of the mean error, first of all, because the same mean value may result from very different distributions of the error along the sequence, err(k). Furthermore, the value of indices $\tilde{k}$ such that $err(\tilde{k})$ is maximum should be also taken into account, since very different results are found if the maximum error occurs in the linear region or at the "knee" of the hysteresis loop, or again at the saturation region. It must be further noticed that each run of the training algorithm, for a given number of epochs, is independent and the weights and biases finally obtained are generally relative to a local minimum of the error function. A common way to determine the optimum network consists in launching the training algorithm multiple times, checking the progression of the error function and comparing the network response with the curves of the training set. The final values of the weights and biases are found after an arbitrary large number of training-test stages. The limits and the critical aspects of this empirical approach are further exacerbated if the model presents a feedback loop, like in our case. It turns out that the network at the "closed loop" configuration (for which the actual and the old values of m at the input are those computed by the FFNN itself in the previous steps) is potentially unstable. In order to speed up and to improve the identification of the optimum network, we have developed a dedicated training procedure, in which the algorithm is launched automatically by an iterative routine. At each iteration, after that the weights and biases are determined by the algorithm, a series of multiple verifications, including the stability check, are made. The success of any verification allows to proceed with the successive one, while the failure is sufficient to immediately repeat the training, thus starting a new iteration. The

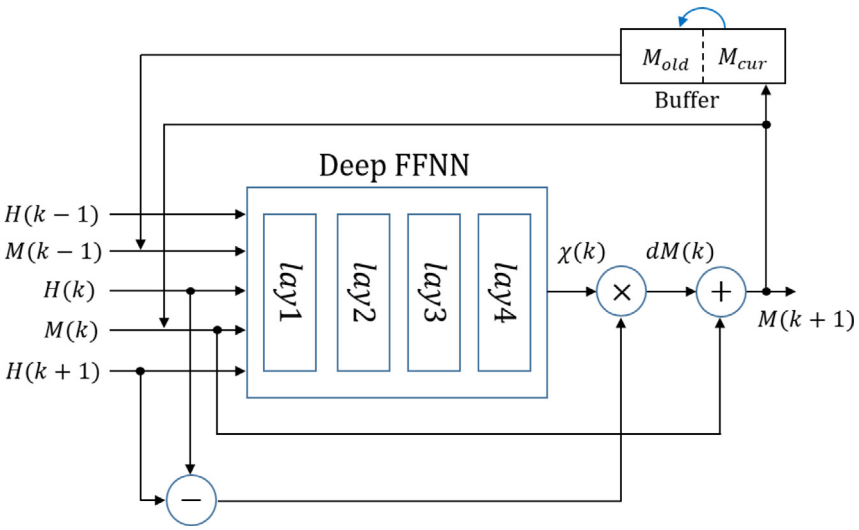



**Fig. 2.** Functional block diagram representing the NN-based hysteresis model.

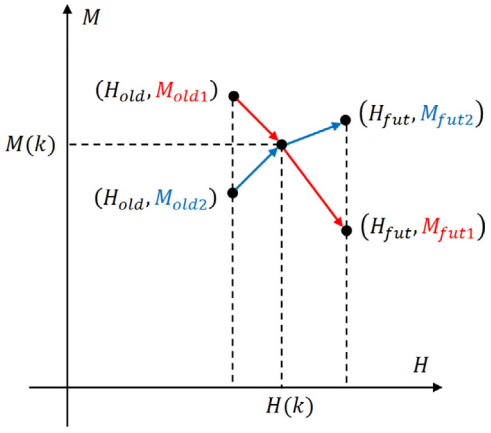



**Fig. 3.** Illustration of the past history dependence in the NN-based model: from the point H(k), M(k) possible future states can be reached depending on the old M.

first verification made is based on the evaluation of the MSE returned by the algorithm, which is compared with an error threshold $E_{TH}$ set by the user. If the MSE is smaller than the threshold, the training set is simulated in "closed loop" configuration and performances are again checked. The MSE is calculated from the simulation results and it is compared with the one returned by the algorithm, that has been obtained in "open loop" configuration. The model is considered stable if the MSE at the "closed loop", $MSE_{CL}$, is smaller than an error obtained by multiplying a certain parameter $\lambda$, set by user, by the MSE of open loop configuration, $MSE_{OL}$. Indeed, the difference between open and close loop is that the second one is affected by the error of the NN output and it could make the solution divergent. Since training the NN in open loop is easier, then it is necessary to continue training the NN in close loop, starting from the one obtained in open loop, and hope that the error at the output is smaller than a certain parameter $\lambda$ set by user. The final verification consists in the computation of the area on the m-h plane enclosed by the hysteresis curve, computed by the FFNN in "closed loop" configuration, and the comparison with the one obtained by the training set. If the relative percentage error between the areas of the training set loop, $err_{A\%}$, is smaller than the threshold $err_{A\%,TH}$, the network is saved and the routine terminates. The flow of operations performed by the computer program for the optimum network identification is represented in Fig. 5.

The training procedure has been utilized at first for the identification of the best network architecture, the number of hidden layers and the number of neurons per each layer. The limits of the under- and over-training have been also experimentally determined. From this preliminary analysis we have concluded that the optimum architecture, relatively to the training set adopted, should be based on 4 hidden layers. Furthermore, it has been found out that if the number of neurons per each layer is balanced, for the same total number of neurons, the MSE is minimized. Finally, the total number of neurons should be comprised between 36 and 60. We have then defined a FFNN with 4 hidden layers and 12 neurons per layer, for which the total number of neurons is $N_TOT = 4(in) + 12 * 4(hidden) + 1(out) = 53$. The transfer function of the neurons belonging to the input layer and the hidden layers is the hyperbolic-tangent sigmoid, defined as

$$f_{tr}(x) = 2/(1 + e^{-2x}) - 1 \tag{7}$$

The 4 input neurons have unitary weights and are unbiased. According to the developed implementation at low level of abstraction, the weights of the hidden neurons are represented as a matrix $LW_j$ for each layer j = 1,2,3,4. The dimensions of $LW_j$ are: $12 \times 4$ for j = 1, $12 \times 12$ for all the others. In practice the number of rows coincides with the number of neurons in the layer, whilst the number of column coincides with the number of synaptic connections of each neuron, which are equal to the number of neurons of the previous layer. On the other hand, to allow the computation of the network output as a series of matrix products, the bias values of the hidden neurons are collected in a series of $12 \times 1$ matrices $LB_j$ (column vectors). Conclusively, the output neuron is represented by the $1 \times 12$ weight matrix OW and a single scalar output bias OB. In this way, the calculation of the network output $y_{NN}$ is very fast:

$$y_1 = f_{tr}(LW_1 * x_{NN}) + LB_1 \tag{8}$$

$$y_j = f_{tr}(LW_j * y_{(j-1)}) + LB_j for j = 2,3,4 \tag{9}$$

$$y_{NN} = f_{tr}(OW * y_4) + OB \tag{10}$$

where $x_{NN}$ is the input vector and $y_j$ is the output of the $j$th hidden layer. The model has a poor memory request, since the neural network is defined by only 541 floating-point variables, which are 492 weights and 49 biases. In addition, the formulation proposed allows an eventual future coupling with FEM solvers based on Coco et al. (2013). This FFNN has been trained according to the procedure previously described, using the parameters listed in Table 3. The obtained weights and bias values can be viewed in Appendix, where the computer programs that implements the NN-based model is shared by the authors.

It is interesting to notice that the proposed methodology can be generally applied also to other network architectures and with different

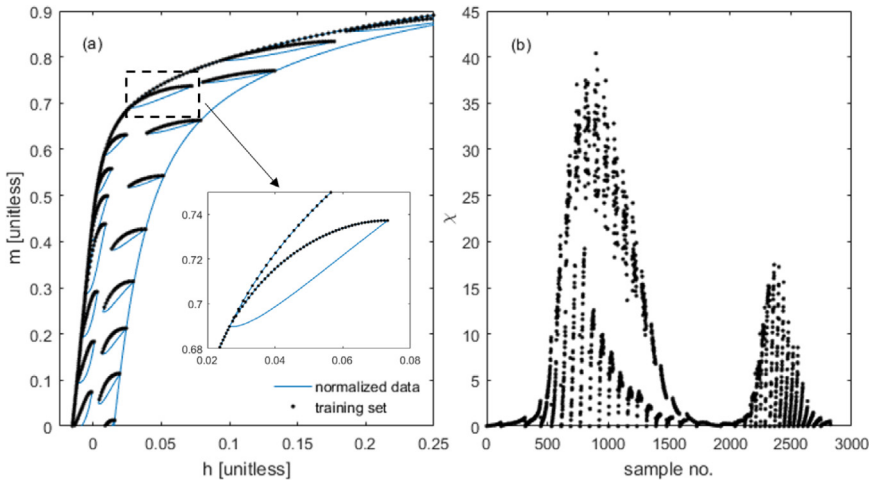



**Fig. 4.** Construction of the training set. Panel (a): Sequences h(k), m(k) plotted on the H-M plane, obtained from the DATASET 1, after the normalization (blue continuous line) and after the successive selection of the points with dh < 0 (black dots). Panel (b): values of the susceptibility applied for the training.

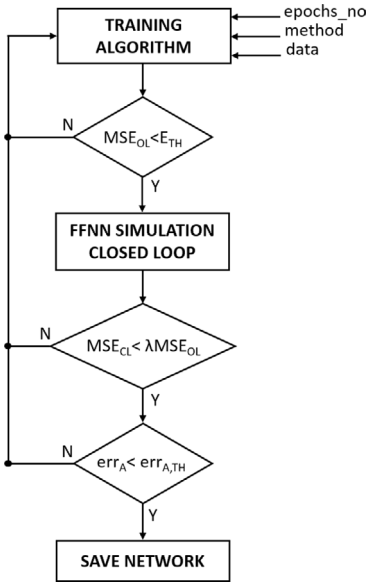



**Fig. 5.** Block scheme representing the operation flow for the optimum neural network identification.

**Table 3**
Parameters of the training program.

| $epochs_{no}$ | Method | $E_{TH}$ | $\lambda$ | err |
|---|---|---|---|---|
| $10^5$ | Scaled-conjugate gradient | 0.8 | 1.5 | 1.5% |

training sets, and that it allows to obtain a network under controlled upper limits of the specifications. Indeed, more strict requirements, respect to those reported in Table 3, could be in principle set, with a consequent increase of the training time.

## 4. Discussion and results comparison

For a better exploration of the possible neural network configurations that had as good as possible simulations in terms of low errors on magnetization prediction, different learning algorithms have been considered. In Table 4 . The best result of the FFNN training, thus, can be summarized by the final value of the mean squared error returned by the scaled-conjugate gradient learning algorithm, $MSE_{OL} = 0.677$, the ratio $MSE_{CL}/MSE_{OL} = 1.29$ and the relative percentage error of

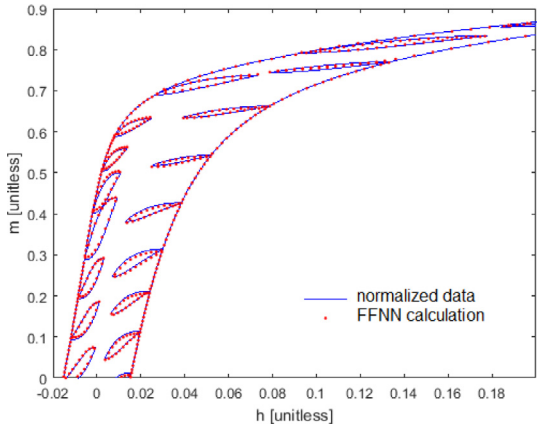



**Fig. 6.** Simulation of the training set curve performed by the FFNN in closed loop configuration and comparison with normalized data.

the loops area $err_{A\%} = 0.69\%$. The effectiveness of the training process is confirmed by the comparison between the training set curve and the one obtained by the FFNN simulation in closed loop configuration applying the same input sequence of the normalized magnetic field. The curves are displayed in Fig. 6.

After the verification of the training accuracy, the NN-based model is finalized by resizing the input and the output to express the magnetic field in Ampere per meter and the magnetization, actually the polarization $\mu_0 M$, in Tesla. Let us firstly examine the capability of the model to generalize the type of hysteresis process already "seen" during the training. The simulation of the saturation loop with some inner loops, in different position respect to those of the training set, is then performed, applying the magnetic field sequence of the DATASET 2. In Fig. 7, the hysteresis loop obtained by the NN-based model is plotted together with the one produced by the Preisach model. A very good agreement has been found in the whole range of the applied magnetic field, even for the minor loops that intersect with each other. This feature is not trivial, since the minor loops of the training set do not have any point in common.

The magnetic losses per unit of volume have been evaluated from the simulated hysteresis loops and the relative percentage error found was equal to 1.67%, confirming the effectiveness and the robustness of the neural network-based model. Another issue that is interesting to consider in neural network modeling is the analysis of the model response when the number of samples of the input sequence is reduced.

**Table 4**
Comparison among different learning algorithms.

| $epochs_{no}$ | Method | $MSE_{OL}$ | $MSE_{CL}/MSE_{OL}$ | Percentage error of the loops area |
|---|---|---|---|---|
| $10^5$ | Scaled-conjugate gradient | 0.677 | 1.29 | 0.69 |
| $10^5$ | Levenberg–Marquardt | 0.777 | 2.05 | 1.45 |
| $10^5$ | SBFGS Quasi-Newton | 0.689 | 1.74 | 0.97 |
| $10^5$ | Resilient Backpropagation | 0.715 | 2.19 | 4.75 |
| $10^5$ | Polak-Ribiére Conjugate Gradient | 0.250 | 12.05 | 14.12 |
| $10^5$ | Resilient Backpropagation | 0.786 | 1.85 | 2.63 |

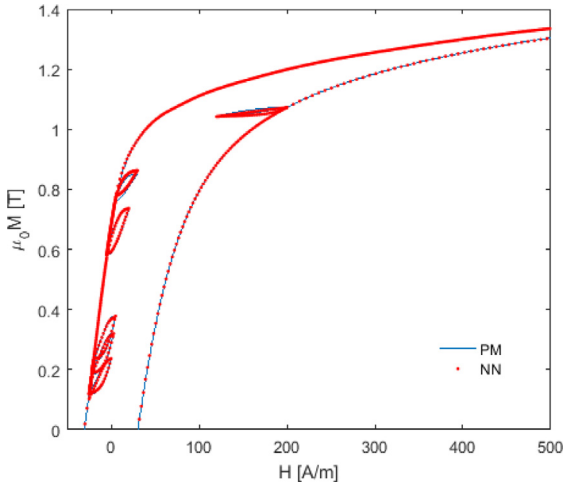



**Fig. 7.** Hysteresis loop of the DATASET 2 computed by the PM is compared with the one obtained by the NN simulation.

Indeed, if two adjacent points are not sufficiently close, the response may be inaccurate or even unstable. For this reason, as second step of the model validation we have examined the network output when the input sequence of the DATASET 2 is under-sampled. The under-sampling factor (USF) was varied between 1 and 6, and the reduced sequences of the magnetic field have been applied to the input of both the PM and the NN. As expected, the curves computed by the PM are practically independent on USF, while those obtained by the NN simulations are progressively getting worse, as one can see in Fig. 8, where the inner loop with inversion point (−5 A/m, 0.59 T) is shown.

However, even if the sequence of the applied magnetic field is rather coarse (for USF = 6, the minor loop shown in the figure consists of only 12 points) the neural network produces a closed curve that is not affected by the typical numerical accommodation phenomenon that occurs in most of the FFNN architectures (Antonio et al., 2021a). The main negative effect is the progressive reduction of the loop area, which leads to an underestimation of the energy loss. The energy losses obtained by evaluating the area of the hysteresis loops simulated by the PM for each value of USF are almost identical, since the undersampling affects only the resolution. Contrarily, the energy loss predicted by the NN is a decreasing function of UFS. The two energy loss curves and their relative percentage error have been plotted against the undersampling factor in Fig. 9.

For $USF < 5$ the reproduction of the energy losses by the NN-model is accurate, being the relative error respect to the values obtained by the PM smaller than 7%. However, for very coarse sequences the loss prediction becomes incorrect, since the error jumps from about 10% to about 30% if USF increases from 5 to 6. After that the capability of the neural network to generalize the training patterns has been successfully verified and the robustness respect to the undersampling has been examined, the next validation step consists in the analysis of the model response to magnetic field sequences the materials are subjected to in practical applications. The simplest working condition that is important to investigate is based on the replication of symmetric hysteresis cycles, taking into consideration different levels of magnetization. The family of loops of the DATASET 3 have then been simulated by the NN-based model. As for the PM simulations, each loop has been traced two times to ensure that the computed curve is closed, but two initial conditions have been investigated, starting from either one vertex or the origin of the H-M plane. It has been found out that the same stable hysteresis loop is produced, regardless the choice of the starting point for the magnetic field and the magnetization. This feature should not be taken for granted since neither the first magnetization curve, nor the branches of any inner symmetric loops, have been taken into consideration for the model training. The comparison between the symmetric hysteresis loops simulated by the neural network and those obtained by the PM computations is shown in Fig. 10 with the related extreme values of the magnetic field applied.

If $H_{extr} < 100$ A/m the extreme value of the magnetization computed by the NN is slightly smaller than the one produced by the PM. The loop branch is less smooth and its slope at the remanence is also a little lower. The coercivity of the inner loops is overestimated, but the function HC(Hextr) is always increasing, i.e. the loops are nested without any intersection. On the other hand, for higher values of the magnetic field applied the accuracy of the NN-based model increases and the magnetization curves are practically coincident with the ones simulated by the Preisach model. The relevant quantities derived from the major loop, such as the coercivity, the remanence and the maximum susceptibility, are correctly reproduced. The energy losses per unit of volume, displayed in Fig. 11, are in this case evaluated for each symmetric loop and plotted against the peak value of the magnetic polarization. The comparison indicates a substantial agreement, since the maximum absolute error found was just of 18 J per $m^3$.

In many power electronics applications, the magnetic materials are subjected to hysteresis processes which are more complex than the symmetric hysteresis loops. For instance, DC biased magnetizations are found in devices supplied by semiconductor-based power supply, small-signal transformers and filtering inductors. Therefore, the test of the NN model also covers the reproduction of magnetization loops in presence of DC bias. The magnetic field sequences defined by the DATASET 4 have been applied as input to the NN and the predicted hysteresis loops have been compared with the ones obtained by the PM simulations. A quite good agreement has been found in all the cases, having different value of the AC components, as it can be viewed in Fig. 12. The loss prediction, shown in Fig. 13, is also noticeably effective.

As last example of the comparative analysis discussed in the paper, the reproduction of a demagnetization pattern, obtained by a cosine wave with exponentially-decreasing amplitude, is here illustrated. This type of hysteresis process may be found during transients of can be exploited to bring the magnetic devices to the unmagnetized state. In the latter case, the way the peak value of H is reached does not matter, whilst the decreasing from the peak to 0 has to be opportunely designed. The magnetic field sequence of the DATASET 5 is shown in the left panel of Fig. 14, while the comparison between the polarization sequences simulated by the PM and the NN-base model is shown in the right panel. During the first four periods moving from the peak value, the magnetic field drops from 1000 A/m to about 136 A/m, and the calculated sequences of the polarization are practically overlapping with each other. During the successive four periods, at the end of which the magnetic field is equal to 18 A/m, the magnetization calculated by

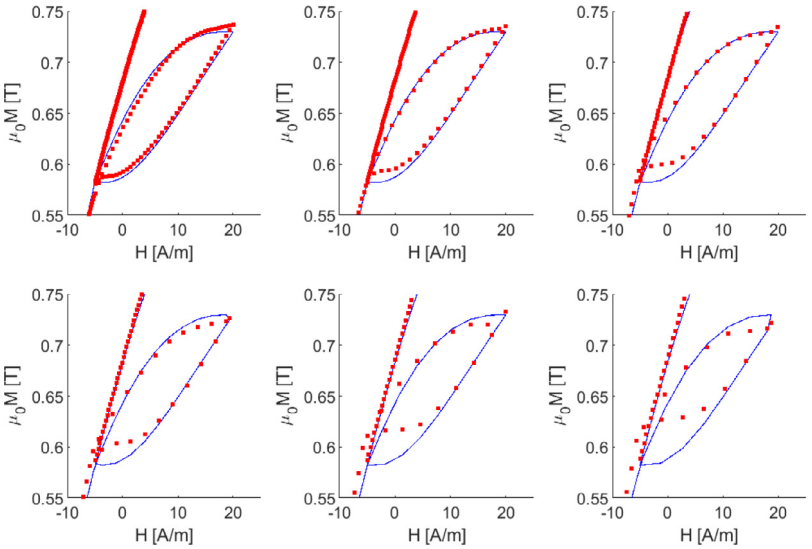



**Fig. 8.** DATASET 2: minor loop along the descending branch at 0.6 T simulated by both the PM and the NN-model for different values of USF.

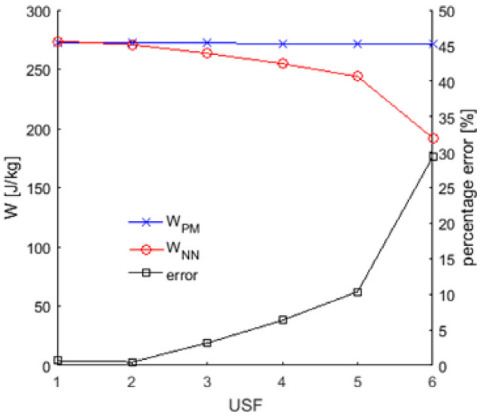



**Fig. 9.** Plot of the energy losses and relative percentage error as a function of the undersampling factor relative to the simulation, performed by both the PM and the NN-model, of the hysteresis loop of the DATASET 2.

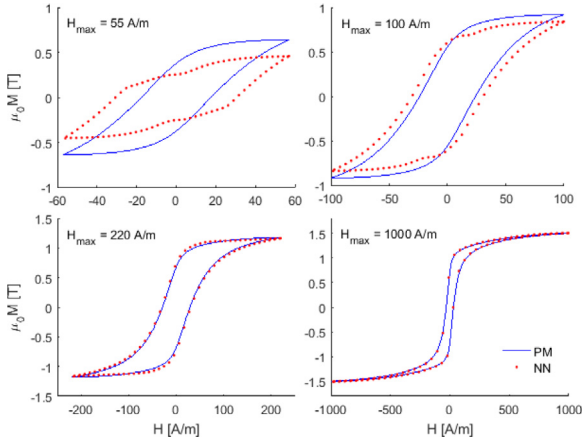



**Fig. 10.** Symmetric hysteresis loops of the DATASET 3 simulated by both the PM and the NN-based model for different extreme values of the applied magnetic field.

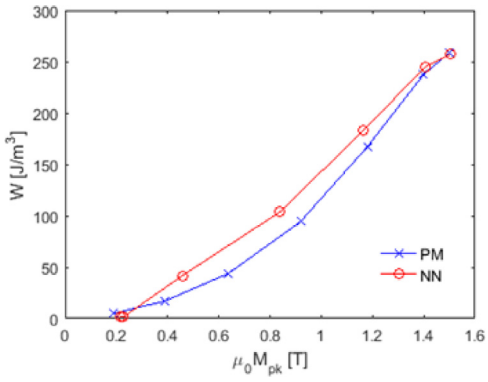



**Fig. 11.** Energy losses obtained by both the PM and the NN simulations versus the peak values of the magnetic polarization.

the NN tends to decrease more rapidly than the one obtained by the PM, with a maximum absolute deviation close to 0.2 T. In the final two periods the agreement between the simulated waveforms is recovered.

The computational cost of the FFNN architecture described here, and the amount of RAM memory occupied, can be expressed in terms of the number $N$ of hidden neurons, equal to 48 in our case, as indicated in Antonio et al. (2021a) and Corti et al. (2019). The low level calculation of the network output requires $N/4(9N+5) = 492$ sums and products, plus $N$ calls to the neuron activation function. The variables of the FFNN to be stored in memory are totally 541 floats subdivided into $N/4(9N + 5)$ weights and N + 1 biases. On the other hand, the computation of the output of the PM, according to the formulation used here, requires the same number $N = 10^6$ (that now indicates the number of hysterons) of sums, products, calls to the weight function, plus 2N threshold comparison, which are the most expensive operations at low level of abstraction. The intrinsic variables of the PM to be stored in memory are 3N floats, subdivided into $N$ values of $H_i$, $N$ values of u and $N$ values of the previous magnetic state. Although the computational complexity of the neural network-based model is $o(N^2)$, while it is linear with $N$ for the Preisach model, it is much faster because the value of $N$ required to obtain smooth hysteresis loop in the excitation range considered is much smaller (49 total neurons against

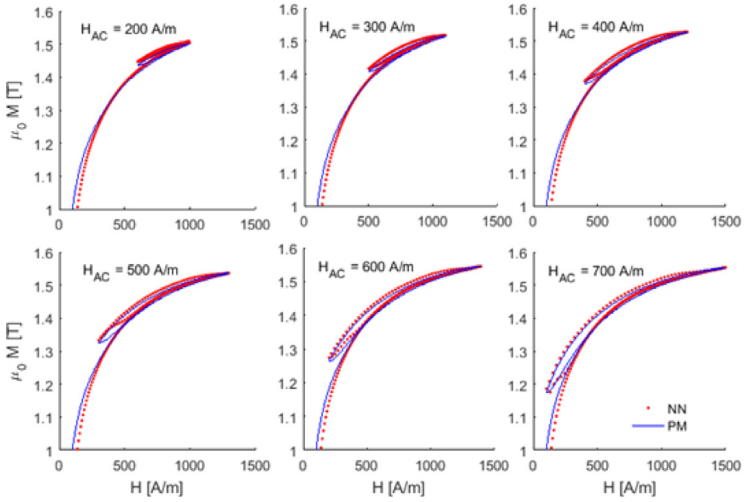



**Fig. 12.** DC biased magnetization loops obtained with HDC = 800 A/m and different amplitudes of the AC component: comparison between PM and NN simulations.

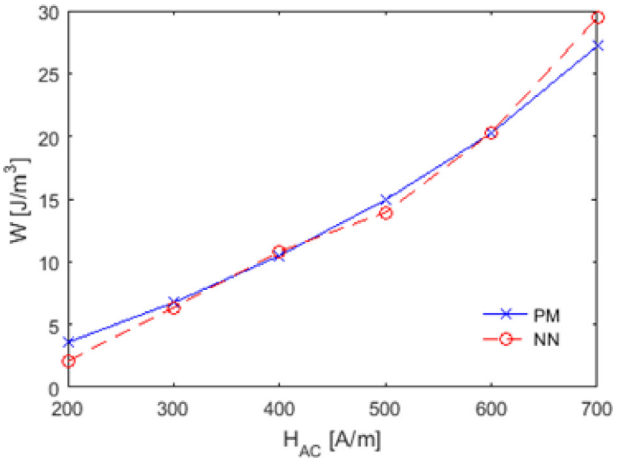



**Fig. 13.** Energy losses calculated by both the PM and the NN as a function of the amplitude of the AC magnetic field component.

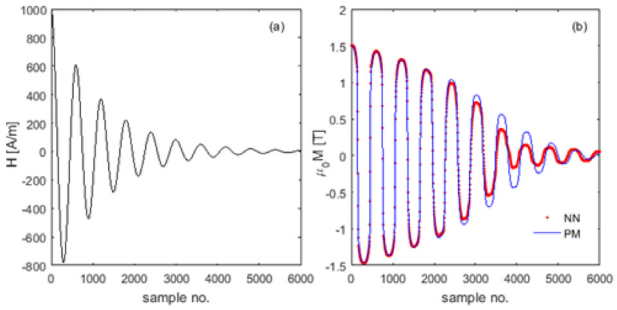



**Fig. 14.** Simulations of the DATASET 5: magnetic field sequence (a) and magnetic polarization sequences (b) calculated by both the PM and the NN-based model.

106 hysterons). In addition, for each Preisach operator two threshold comparisons must be performed, whilst the artificial neurons only require sums and multiplications. The computational speed estimated for the PM is 31.2 samples/s, while for the NN 29.7 ksamples/s, about three orders of magnitude higher. All the simulations described in the paper have been performed by the same computer, equipped with a CPU Intel® Core™ i7–2670QM @ 2.20 GHz, 8 GB of RAM memory, under a 64-bit operating system.

## 5. Conclusions

A novel approach to model hysteresis processes for soft ferromagnetic materials based on a "standalone" deep neural network has been presented. The architecture of the model has been designed to account for the past history dependence, allowing the reproduction of hysteresis loops under generic supply excitations without the need to couple the neural network with other models or algorithms. A suitable training procedure, based on an iterative algorithm, helps the identification of the network parameters under fixed performances. The Preisach model, that can be easily identified starting with the minimal material data usually provided by the manufacturers, has been adopted here to generate an appropriate training set and other generic hysteresis processes for the model validation. The neural network-based model turns out to be able to replicate the magnetization loops obtained by the PM and to predict the energy losses with acceptable accuracy in the entire range of excitations. Furthermore, the implementation of the model at low level of abstraction led to a considerable gain in terms of simulation time, about three orders of magnitude lower respect to the PM (also implemented at low level). The great computational and memory efficiency of the model (i.e. little memory used and low computational costs for each trained DNN) makes it suitable for future matching with finite element analysis. Indeed, as is well known, finite element analysis requires many nodes in which the fields must to be calculated and each node needs of a hysteresis model: in our case, each node may use a trained DNN.

## CRediT authorship contribution statement

**Simone Quondam-Antonio:** Investigation, Formal analysis, Data curation, Software, Writing – original draft. **Francesco Riganti-Fulginei:** Supervision, Conceptualization, Investigation, Writing – review & editing. **Antonino Laudani:** Supervision. **Gabriele-Maria Lozito:** Supervision. **Riccardo Scorretti:** Supervision.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Data availability

No data was used for the research described in the article.

## Appendix

The source code relative to the main computer program and the function that implements the NN model, are here shared by the authors. The main program starts with the definition of the sequence of the applied magnetic field, that has a triangular shape between — Hmax and Hmax. The sequence starts with a first magnetization curve, to reach the value of Hmax from the un-magnetized state, and then the triangular wave is replicated for np periods. The magnetization is processed at sample step, calling the function **Nnmodel_fast.m**, and stored in the vector $M_{nn}$. At the end of the iterative loop the computational speed is evaluated and displayed in the screen. Both the sequences of the magnetic field and the magnetization are truncated to the last period and the area enclosed by the resulting hysteresis loop is computed and displayed to the screen. Conclusively, the loop is plotted and the sequences of H and M are saved in ".dat" format.

```
% Main program for the simulation of hysteresis loops with DeepNetwork Model
% Program written by Simone Quondam Antonio on 30th April 2021

clear all; close all; clc;

opengl software

% Define input magnetic field sequence and initialise matrices %
H_max = 100;      % maximum magnetic field [A/m]
np = 2;      % number of periods
SPP = 600;      % samples per period
H_des = linspace(H_max,-H_max+4*H_max/SPP,SPP/2)'; % descending branch
H_asd = linspace(-H_max,H_max-4*H_max/SPP,SPP/2)'; % ascending branch
H_loop = [H_des;H_asd]; H_loop = repmat(H_loop,np,1);
H_fmc = linspace(0,H_max-H_max/100,100)'; % first magnetization curve
H_test = [H_fmc;H_loop];   % add first magnetization curve
clear H_loop H_asd H_des H_fmc  % release memory
max_points_test = size(H_test,1);
M_nn = zeros(size(H_test));  % initialization of output matrix

M_nn(1) = 0; % first sample (unmagnetised state)

% Sample-by-sample computation %
tic
for ct=2:max_points_test-1
dH = H_test(ct+1)-H_test(ct);
[M_nn(ct+1)] = NNmodel_fast(dH,H_test(ct),M_nn(ct),H_test(ct-1),M_nn(ct-1));
end

CS = (max_points_test-1)/toc;
disp(strcat('Computational speed: ',num2str(CS),' samples/s'));
clear CS;

% Selection of last loop period %
M_nn = M_nn(end-SPP+1:end);
H_test = H_test(end-SPP+1:end);

% Calculation of loop area %
AloopNN = polyarea(H_test,M_nn)
disp(strcat('Loop area = ',AloopNN,'J/m^3'));

% Plot hysteresis loop %
figure(1);
plot(H_test,M_nn,'-');
xlabel('H [A/m]'); ylabel('\mu_0M [T]');

% Save in and out vectors to .dat file %
fidH = fopen('H_nn','a')
fprintf(fidH,'%d\n',length(H_test));
for(i = 1:length(H_test));
    fprintf(fidH,'%f\n',H_test(i));
end
fclose(fidH);

fidM = fopen('M_nn','a')
fprintf(fidM,'%d\n',length(M_nn));
```

```
for(i = 1:length(H_test));
    fprintf(fidM,'%f\n',M_nn(i));
end
fclose(fidM);
```

The function **NNmodel_fast** is called at each step to solve the future value of the magnetization Mfut as a function of the actual and the previous values of both H and M, and the actual increment dH. In order to speed up the simulations, the weights and the bias values which characterize the trained NN-based model are defined locally in the function.

```
%% NN-based model: implementation at low level of abstraction %%
%% INPUTS: IN = [dH,H,M,Hold,Mold] %%
%% OUTPUT: Mfut (future value of M) %%
%% Developed by Simone Quondam Antonio on 29th April 2021 %%

function [Mfut] = NNmodel_fast (dH,H,M,Hold,Mold);
Hmax = 2000; Mmax = 1.590; %% Scaling
in_min_map = [-0.9883,-0.9994,-0.9883,-0.9994];
in_max_map = ones(size(in_min_map));
out_min_map = 0.0030;
out_max_map = 33.2705;
IN = zeros(4,1); % Input vector
if(dH < 0);
   IN = [H/Hmax,M/Mmax,Hold/Hmax,Mold/Mmax]';
else
   IN = [-H/Hmax,-M/Mmax,-Hold/Hmax,-Mold/Mmax]';
end

%% Normalization of input vector %%
IN_norm = zeros(size(IN));
for(i = 1:length(IN));
   IN_norm(i) = 2*(IN(i)-in_min_map(i))/(in_max_map(i)-in_min_map(i))-1;
end

%% Bias values %%
    HB = [
    2.8255   -1.8326    1.5807    1.8762   % neuron 1 lay 1 to 4
   -3.1997    1.2556    2.1759    0.9550   % neuron 2 lay 1 to 4
   -1.6598   -1.0174    1.1006   -0.9980
   -0.1057   -0.9303    0.8733    1.4184
   -1.8805    0.1419   -0.0004    1.2312
   -0.1627    0.0320   -0.2732   -0.3987
   -0.0963   -0.0295    0.0594   -0.3758
    0.3058   -0.5342   -0.2539    0.1387
    1.4029    1.0549   -0.8857    0.2458
   -1.7912    0.9910   -0.9196    0.9346
   -2.1124    1.3677    0.8981    1.4900
    2.3198    1.7903   -1.8867    2.0456];
    OB = 1.6941834e+00; % bias of the output neuron

%% Neuron Weights %%
% Hidden LAY 1: 12 neurons with 4 weights %
    LW1 = [
    -1.7056    1.7137    0.2698    0.5109  % neuron 1
    0.6873   -1.8701   -0.9129   -2.3873   % neuron 2
    1.0553    0.0908    0.8318    2.7299   % ...
   -2.5534    0.2790   -3.4273   -1.9708
    0.2309   -0.1816   -1.7513   -1.6246
    2.3944    0.8186    3.4641   -2.2951
   -2.5739    0.8953   -4.8557   -0.8050
    1.3825    1.6555    1.8037   -1.7973
    1.9804   -2.5189    0.6911    0.0233
   -1.6130    1.6311    1.5672    0.8540
   -0.9159   -0.5805    2.2727    0.6202
    1.1119   -0.8285   -1.9111   -1.3061]; % neuron 12

% Hidden LAY 2: 12 neurons with 12 weights %
    LW2 = [
```

```
    0.1958    0.4285   -0.3212   -1.0349   -0.6069    0.3770   -0.6538    0.5905
       0.0964    0.6630   -0.2877   -0.4846
   -0.8554   -1.4324    0.6909    0.6460   -0.9497   -0.5832    0.2866   -0.2655
      -0.3076    0.4295    0.5345   -0.0216
    0.7003   -0.7307    0.8311   -1.3918   -0.2911    0.4770   -0.1628    0.1218
       -0.9451    0.0693   -0.2421   -0.6417
   -0.0488    0.5699    0.3455    0.7820    0.0698    0.1786   -0.4111   -0.7473
      0.3064    0.7202    0.2608   -0.5728
   -0.4203    1.0835   -0.5066    0.9134    0.0672   -2.7198    2.3125   -2.0815
      0.8676   -1.0476    0.3197    0.2823
    0.5527   -0.0172    0.9353   -0.5655   -0.4336   -0.4412   -0.3444   -0.2715
       -1.1957    0.9722    0.3348   -0.6194
   -0.0212    0.6749    0.2675   -2.3715   -0.5489    2.3950   -2.9587    1.1929
      1.2529    0.4968    0.2996   -0.4387
   -0.0976   -0.5852   -0.8828    1.5270   -0.9375   -2.0251    1.5456   -0.5112
      -1.2609    0.0913    0.4746   -0.7330
    0.7721    0.7316   -0.0339   -0.3454    1.1929    1.9022   -1.0431   -0.1632
       0.3388   -0.8358    0.4739    0.3188
    0.6417   -0.7394   -0.0171   -0.4743   -0.5231    0.0653   -0.8503   -0.2327
       0.3854    0.8315    0.2352   -0.0246
    0.2450    0.3380   -0.8824   -1.4678    1.0958    0.4850   -1.5172   -1.2754
       0.7105   -0.7690    0.8216    0.2520
    0.2602   -0.4691    1.6366   -0.4038    0.5863   -0.3638    1.6298   -0.3243
       0.2781    0.0440    0.5810    0.4117];
% Hidden LAY 3: 12 neurons with 12 weights %
    LW3 = [
    -0.3285    1.1156    0.6798    0.1839   -1.1876   -0.4033   -0.1448   -0.5414
       -1.2473   -0.2880   -0.8320    0.2037
    0.2241    0.8620    1.1002   -0.9206    2.9690    0.4312   -3.5206   -0.4567
       -0.2767   -0.1644   -0.1363    1.8558
   -0.0661   -0.5449   -0.3549   -0.5132   -0.5800   -0.4233   -0.5669    0.5229
      -0.2346    0.5540    0.5054    0.2982
   -0.6671   -0.2867   -0.3773   -0.1886    0.3477   -1.2085    0.7003   -0.4778
      0.1478    0.2861    1.1715    0.9932
   -0.3830   -1.0794   -0.3597    0.0962    1.0783   -0.1105   -3.2098    1.4509
      -2.2032   -1.0649   -1.4903   -0.6602
    0.4287   -0.7779    0.0314    0.5469   -1.2519   -0.4021    0.8197   -0.1429
       0.7765   -0.4858    0.5102   -1.2608
    0.6393    0.9519   -1.2447    0.7010   -0.9377   -0.1940   -1.2089    0.2252
       -0.4808    0.1347   -0.8428    0.4357
   -0.5079   -0.7960    0.8791   -0.2367   -0.3378   -0.9301    2.2311   -1.8267
      -0.4956    0.5105    0.5960   -0.5383
   -0.0299   -0.4856    0.0442   -0.3706    2.1319   -0.6284   -1.3033   -1.3416
      -0.2204   -0.7586    0.7628    0.0853
   -1.0812    0.2882   -1.2577    0.1397    0.3612   -0.2256   -1.2300    1.9147
      0.0897   -0.6627   -0.6614   -0.1379
    0.8943    0.2945    0.1349    0.1647    0.5166    1.2561   -1.1566    0.6283
       -0.5368    0.4687   -0.8601    0.2680
    0.0452    0.7521   -0.4944   -0.1205    1.8067    1.3554   -2.0949    0.9744
       -0.9474    0.8894   -2.3356    0.5771];

% Hidden LAY 4: 12 neurons with 12 weights %
    LW4 = [
    -0.3434   -0.1178    0.3471   -0.5749    0.3302   -0.1977    0.3508   -1.1425
       -0.8719    0.9148   -0.9428   -1.3124
   -1.5154   -0.5008   -0.6542   -0.8312    1.1576   -0.7050    0.3418    1.5964
      -0.9919   -2.0236   -1.6925   -0.1587
    1.3749   -0.7816    0.2230   -0.7894   -1.0885   -0.7719    0.6102   -0.7602
       -1.1155    0.2368   -0.9731    0.6548
   -0.8081   -0.0050    0.4411    0.8400   -2.7981   -0.7577   -0.0178    0.7388
      -0.7345   -1.3307   -1.9633   -2.4118
   -0.4941    0.9496    0.3375    0.0741   -3.4038   -1.3266   -1.6440    1.3793
      -0.0617   -2.3320   -1.7215   -3.6927
    0.4257    0.5231    0.0467   -0.1124    0.0376    0.2832    0.6261    0.2506
       -0.7543    0.2852   -0.5125   -0.1051
```

```
    -1.4292    -0.6672     0.1818    -0.4437     0.2712    -0.4533    -0.5216       -0.6502
       0.3339     0.1007    -0.4520    -0.6287
     0.1381     1.0086    -0.5386    -1.1454     0.4312     0.2992     0.7978     0.7578
        0.3842    -0.3569     1.6854    -0.0933
     0.3227     1.9018    -1.3730     0.3132    -0.8732    -1.1639     0.8515       -0.2337
        1.2468     1.0582    -0.7716     0.2015
    -0.0847     1.0014     0.1531     0.0695    -0.3549     0.2710    -1.0226     0.1968
       1.0787     0.1826    -0.6443     0.8825
     0.5481    -1.5357     0.0357     0.6214    -0.7136     0.2061     0.5887       -1.8545
        0.2530     0.1751     0.4965    -0.3849
     0.1723    -2.4616    -0.0961     0.5424     0.1351     0.6888    -0.4049     0.2261
        -0.8002     0.6786     0.4581     0.2621];

% Out LAY: weights of the output neuron %
     OW = [
     0.9586     1.9879     0.8295    -1.5228    -2.5074    -1.1458    -0.3219     1.0737
        -1.5995     1.3707     2.1713    -2.1054];

%% Compute NN output %%
inop = tansig(LW1*IN_norm+HB(:,1));
outop=purelin(OW*tansig(LW4*tansig(LW3*tansig(LW2*inop+HB(:,2))+HB(:,3))+...
HB(:,4))+OB);

chi = (outop+1)/2*(out_max_map-out_min_map)+out_min_map; % de-normalise chi
Mfut = (chi*Mmax/Hmax)*dH + M; % future value of mu0*M in [T]
end
```

## References


Adly, A., Abd-El-Hafiz, S., 1998. Using neural networks in the identification of Preisach-type hysteresis models. IEEE Trans. Magn. 34 (3), 629–635.

Adly, A.A., Abd-El-Hafiz, S.K., 2014. Utilizing neural networks in magnetic media modeling and field computation: A review. J. Adv. Res. 5 (6), 615–627.

Antonio, S.Q., 2019. Optimum identification of iron loss models in NGO electrical steel for power electronics. In: 2019 IEEE 5th International Forum on Research and Technology for Society and Industry (RTSI). IEEE, pp. 182–187.

Antonio, S.Q., Fulginei, F.R., Laudani, A., Faba, A., Cardelli, E., 2021a. An effective neural network approach to reproduce magnetic hysteresis in electrical steel under arbitrary excitation waveforms. J. Magn. Magn. Mater. 528, 167735.

Antonio, S.Q., Ghanim, A., Faba, A., Laudani, A., 2021b. Numerical simulations of vector hysteresis processes via the Preisach model and the Energy Based Model: An application to Fe-Si laminated alloys. J. Magn. Magn. Mater. 539, 168372.

Cardelli, E., Faba, A., Laudani, A., Antonio, S.Q., Fulginei, F.R., Salvini, A., 2016a. A moving approach for the Vector Hysteron Model. Physica B 486, 92–96.

Cardelli, E., Faba, A., Laudani, A., Lozito, G., Fulginei, F.R., Salvini, A., 2016b. A neural-FEM tool for the 2-D magnetic hysteresis modeling. Physica B 486, 111–115.

Chen, G., Chen, G., Lou, Y., 2021. Diagonal recurrent neural network-based hysteresis modeling. IEEE Trans. Neural Netw. Learn. Syst..

Cirrincione, M., Miceli, R., Galluzzo, G.R., Trapanese, M., 2002. Preisach function identification by neural networks. IEEE Trans. Magn. 38 (5), 2421–2423.

Coco, S., Laudani, A., Fulginei, F.R., Salvini, A., 2013. Neural-FEM approach for the analysis of hysteretic materials in unbounded domain. COMPEL.

Corti, F., Grasso, F., Paolucci, L., Pugi, L., Luchetti, L., 2019. Circular coil for EV wireless charging design and optimization considering ferrite saturation. In: 2019 IEEE 5th International Forum on Research and Technology for Society and Industry (RTSI). IEEE, pp. 279–284.

Fulginei, F.R., Lozito, G.M., Gaiotto, S., Salvini, A., 2015. Improving the Jiles-Atherton model by introducing a full dynamic dependence of parameters. In: 2015 IEEE 1st International Forum on Research and Technologies for Society and Industry Leveraging a Better Tomorrow (RTSI). IEEE, pp. 161–165.

Fulginei, F.R., Salvini, A., 2005. Softcomputing for the identification of the Jiles-Atherton model parameters. IEEE Trans. Magn. 41 (3), 1100–1108.

Hussain, S., Lowther, D.A., 2017. The modified Jiles–Atherton model for the accurate prediction of iron losses. IEEE Trans. Magn. 53 (6), 1–4.

Jacques, K., Steentjes, S., Henrotte, F., Geuzaine, C., Hameyer, K., 2018. Representation of microstructural features and magnetic anisotropy of electrical steels in an energy-based vector hysteresis model. AIP Adv. 8 (4), 047602.

Jiles, D.C., Atherton, D.L., 1986. Theory of ferromagnetic hysteresis. J. Magn. Magn. Mater. 61 (1–2), 48–60.

Kuczmann, M., Ivanyi, A., 2002. A new neural-network-based scalar hysteresis model. IEEE Trans. Magn. 38 (2), 857–860.

Laudani, A., Fulginei, F.R., Salvini, A., 2014. Comparative analysis of Bouc–Wen and Jiles–Atherton models under symmetric excitations. Physica B 435, 134–137.

Laudani, A., Lozito, G.M., Riganti Fulginei, F., Salvini, A., 2015. On training efficiency and computational costs of a feed forward neural network: a review. Comput. Intell. Neurosci. 2015.

Leite, J.V., Benabou, A., Sadowski, N., 2009. Accurate minor loops calculation with a modified Jiles-Atherton hysteresis model. COMPEL.

Leite, J.V., Sadowski, N., Da Silva, P.A., Batistela, N.J., Kuo-Peng, P., Bastos, J.P., 2007. Modeling magnetic vector hysteresis with play hysterons. IEEE Trans. Magn. 43 (4), 1401–1404.

Lin, D., Zhou, P., Rahman, M., 2017. A practical anisotropic vector hysteresis model based on play hysterons. IEEE Trans. Magn. 53 (11), 1–6.

Locorotondo, E., Pugi, L., Corti, F., Becchi, L., Grasso, F., 2019. Analytical model of power MOSFET switching losses due to parasitic components. In: 2019 IEEE 5th International Forum on Research and Technology for Society and Industry (RTSI). IEEE, pp. 331–336.

Longhitano, M.R., Sixdenier, F., Scorretti, R., Geuzaine, C., Krähenbühl, L., 2019. Test-case transformer for the energy-based vector hysteresis model. In: 2019 22nd International Conference on the Computation of Electromagnetic Fields (COMPUMAG). IEEE, pp. 1–4.

Makaveev, D., Dupré, L., De Wulf, M., Melkebeek, J., 2001. Modeling of quasistatic magnetic hysteresis with feed-forward neural networks. J. Appl. Phys. 89 (11), 6737–6739.

Mirzaei, M., Ripka, P., Vyhnanek, J., Chirtsov, A., Grim, V., 2021. Modeling of hysteresis loops using rational and power functions. J. Magn. Magn. Mater. 522, 167563.

Ram, B.S., Kulkarni, S., 2019. An isoparametric approach to model ferromagnetic hysteresis including anisotropy and symmetric minor loops. J. Magn. Magn. Mater. 474, 574–584.

Salvini, A., Fulginei, F.R., Pucacco, G., 2003. Generalization of the static Preisach model for dynamic hysteresis by a genetic approach. IEEE Trans. Magn. 39 (3), 1353–1356.

Sarker, P.C., Guo, Y., Lu, H.Y., Zhu, J.G., 2020. A generalized inverse Preisach dynamic hysteresis model of Fe-based amorphous magnetic materials. J. Magn. Magn. Mater. 514, 167290.

Scorretti, R., et al., 2022. Algorithms to reduce the computational cost of vector Preisach model in view of Finite Element analysis. J. Magn. Magn. Mater. 546, 1–15.

Sixdenier, F., Scorretti, R., Marion, R., Morel, L., 2008. Quasistatic hysteresis modeling with feed-forward neural networks: Influence of the last but one extreme values. J. Magn. Magn. Mater. 320 (20), e992–e996.

Steentjes, S., Henrotte, F., Hameyer, K., 2017. Energy-based ferromagnetic material model with magnetic anisotropy. J. Magn. Magn. Mater. 425, 20–24.

Upadhaya, B., Rasilo, P., Perkkiö, L., Handgruber, P., Belahcen, A., Arkkio, A., 2020. Comparison of anisotropic Energy-based and Jiles–Atherton models of ferromagnetic hysteresis. IEEE Trans. Magn. 56 (4), 1–7.

Viana, A., Coulomb, J.-L., Rouve, L.-L., Cauffet, G., 2010. Numerical resolution of the modified Langevin equation using a differential expression: Application to the Jiles magnetostriction law of approach. J. Magn. Magn. Mater. 322 (2), 186–189.