Modeling electrical properties for various geometries of antidots on a superconducting film

Sajjad Ali Haider¹ · Syed Rameez Naqvi¹ · Tallha Akram¹ · Muhammad Kamran¹ · Nadia Nawaz Qadri¹

Received: 28 September 2017 / Accepted: 9 November 2017 / Published online: 17 November 2017
© The Author(s) 2017. This article is an open access publication

Abstract Electrical properties, specifically critical current density, of a superconducting film carry a substantial importance in superconductivity. In this work, we measure and study the current–voltage curves for a superconducting Nb film with various geometries of antidots to tune the critical current. We carry out the measurements on a commercially available physical property measurement system to obtain these so-called transport measurements. We show that each of the used geometries exhibits a vastly different critical current, due to which repeatedly performing the measurements independently for each geometry becomes indispensable. To circumvent this monotonous measurement procedure, we also propose a framework based on artificial neural networks to predict the curves for different geometries using a small subset of measurements, and facilitate extrapolation of these curves over a wide range of parameters including temperature and magnetic field. The predicted curves are then cross-checked using the physical measurements; our results suggest a negligible mean-squared error—in the order of 10^{-9} .

Keywords Critical fields · Transport properties · Critical currents · Vortex pinning · Artificial neural networks

Introduction

Various geometries of artificial pinning centers with vortex lattice have been investigated to observe the behavior of vortex motion and pinning interaction (Baert 1995; Cuppens et al. 2011; He et al. 2012; Jaccard et al. 1998; Kamran et al. 2015; de Lara et al. 2010; Latimer et al. 2012; Martin et al. 1999, 1997). All these studies have resulted in improving the current carrying properties of superconducting materials for use applications. The critical current density is one of the most important properties of superconductors. The superconducting film with an array of artificial pinning centers by nanolithography techniques can enhance the current density. If each defect accommodates one flux, then the current density increases and resistance decreases, but as the vortex lattice becomes disorder, the current density decreases and resistance increases. In this way, the current voltage (IV) characteristics or curves of the pinning centers on a superconducting film show abrupt changes. Such abrupt changes give the IV curves measurement, also called the transport measurement, a substantial prominence in the field of superconductivity.

During our experimental work, it has been recognized that transport measurements are notorious and cumbersome to measure, especially when they are repeatedly needed (Kamran et al. 2016). We believe that an approximation model, such as one based on artificial neural networks (ANN) proposed in this work, can help in evading the repeated measurements, and rather extrapolate the curves using their smaller subset for unforeseen parameters. The proposed methodology not just relieves the researchers from this tedious procedure—saving them time, cost, and energy, but is applicable to various geometries of antidots—giving this study a significant importance.

ANN are said to be mathematical equivalents of a human brain that tend to learn in one situation, and repeat in another

✉ Syed Rameez Naqvi
rameeznaqvi@ciitwah.edu.pk

¹ Department of Electrical Engineering, COMSATS Institute of Information Technology, G. T. Road, Wah Cantt, Pakistan

(Guojin et al. 2007). They are supposed to recognize patterns, and establish relationships between independent variables of a smaller subset of values. These relationships are then used in solving problems that need approximation and prediction for a larger sample space having unforeseen values, where the ANN are typically found extremely useful (Cybenko 1989). ANN may simply be described as a network of interconnected nodes, each having some weight and bias—called network coefficients. The purpose of such a network is to provide mapping between inputs and outputs, where the latter are obtained after a successful learning phase. The process of learning comprises training and validation. In the training phase, several pairs of inputs and outputs are iteratively provided to the network to enable the ANN to establish a mathematical relationship. The coefficients are updated in each iteration until the network converges to an optimal solution. The manner, in which these coefficients are updated, distinguishes one *training* algorithm from the rest (Hornik 1991, 1993; Hornik et al. 1989).

While the ANN have been widely ratified in engineering applications (Elminir et al. 2007; Ghanbari et al. 2009; Reyen et al. 2008), their use in the field of material science, especially for the purpose of modeling electrical properties, is inordinately constrained (Haider et al. 2017; Kamran et al. 2016; Quan et al. 2016; Zhao et al. 2016). In this work, we explore the prediction of IV curves of a superconducting film with various geometries of antidots using three commonly used ANN architectures, trained by various learning algorithms. The predicted results are then compared with the actual measurements; this step is termed validation. Although slightly different from each other, our approximated results from each training algorithm achieve high accuracy—smaller mean-squared error (MSE). Besides describing the approximation methodology, this work is intended to present a comparison between three ANN architectures, and three training algorithms in terms of prediction accuracy (given in MSE), training time, and number of iterations taken to converge to an optimal solution.

The rest of the article is organized as follows: Section “Physical measurement system and readings” presents detail of the experimental setup to obtain the transport measurements, and a brief commentary on their characteristics. In Section “Artificial neural network model”, we present the approximation methodology based on ANN, followed by the results and discussions in Section “Research methodology and simulation results”. We conclude the paper in Section “Conclusion”.

Physical measurement system and readings

Our experimental setup for transport measurements primarily comprises a physical properties measurement system (PPMS) from quantum design. An Nb film of 16 nm

thickness is deposited on an SiO_2 substrate for obtaining the desired arrays of antidots. The micro bridges and nanostructured array of antidots are, respectively, fabricated by photo- and e-beam lithography techniques on a resistive layer of polymethyl metacrylline (Mostako and Alika 2012; Shahid et al. 2016). Fabrication of the microbridges and array is followed by scanning of the samples to the desired antidots using scanning electron microscopy (SEM). We subsequently mount our sample in the PPMS for transport measurements, which are carried out by four-probe method in PPMS with temperature fluctuation within ± 3 mK, and the external magnetic field applied perpendicular to the plane of the film. We have also swept the field from -8 to 8 mA at constant temperature. During the entire measurement process, which is always carried out in high vacuum, a small amount of liquid helium is placed inside the chamber to prevent overheating.

Figure 1 presents SEM of the four geometries that we investigate in this work, and Fig. 2 corresponds to their respective IV curves measured at different values of temperature. The top left, top right, bottom left, and bottom right in both figures, respectively, correspond to the rectangular, square, honeycomb, and kagome arrays of antidots. These curves may be divided into three regions according to their slopes: in the first region, the voltage is almost zero for gradually increasing current, followed by a sudden jump in the voltage in the second region, and finally, in the third region, there exists a linear relationship between the two variables. Two important observations that may be conveniently made from these figures are:

1. The IV curves show a sudden jump at critical current (I_c) in the second region, which resembles Shapiro steps. These steps usually appear when the interstitial vortex lattice is formed, and due to high vortex velocities, instability may occur; as a result of which the system shows a step.
2. The sharpness in the curves significantly varies for each geometry: those having larger interstitial area may accommodate a larger number of vortices, leading to increased energy conservation in those geometries. Therefore, the honeycomb and kagome arrays will exhibit flatter or smoother curves in comparison with sharp steps for rectangular and square arrays of antidots.

After successively performing the transport measurements, we had obtained a three-dimensional (H , T , I) data set comprising $[4 \times 4 \times 1600]$ values for each film having a different geometry. Note that we have taken out one curve from this data set for each geometry, and kept it isolated from the entire ANN modeling process. These four curves should be used for cross-checking our approach on, so to say, unforeseen data values, once the system had been completely designed using the MATLAB's ANN toolbox

Fig. 1 SEM of rectangular (top left), square (top right), honeycomb (bottom left), and kagome (bottom right) arrays

Fig. 2 Transport measurements at various temperatures and zero magnetic flux

on the modified data set (the one excluding the curves extracted for cross-checking). The toolbox, by default, divides the provided data set into three: the first subset is used for the training purpose, 50% of the remaining values is used for validation, while the rest is strictly kept isolated, which is then used for the testing purposes. The modified data set values were still copious enough to give us confidence in allocating a large data set exclusively for the training purpose. However, while performing the simulations, we realized that increasing the size of training set beyond 70% would not give us a considerable advantage in terms of prediction accuracy. MATLAB's ANN toolbox also uses seventy percent of the data values for the training purpose by default, which further justifies our selection of training and testing data sets. In the next section, we elaborate on the ANN's operation principle, training algorithms, and architectures used in this work.

Artificial neural network model

The structure and operation of a neuron—the basic building block of ANN—have been described on several occasions (Guclu et al. 2015). Briefly, it is a cascaded design, from an input layer to an output layer, where functional blocks are sandwiched either between input and hidden, or hidden and output layers, where each layer may comprise a specific number of neurons. It is believed that the number of hidden layers in a network is directly proportional to the prediction accuracy, i.e., the greater the number of hidden layers, the more accurate the results will be at the expense of complexity. However, problems similar to the one addressed in this work merely require up to two hidden layers for an acceptable accuracy level in most cases (Setti et al. 2014). The mapping between input and output layers, achieved through a few hidden layers, may follow one of the three most widely adopted architectures: feedforward, cascaded, and layer-recurrent neural nets. While the difference between the three architectures will be highlighted later, in what follows we make use of the simplest one, feedforward, just to describe the operation of ANN. Figure 3 depicts a fully connected feed-forward neural net with a single hidden layer (δ_H). R number of inputs are connected to input layer (δ_I) with S number of neurons, whereas outputs generated by input layer acts as a source for the hidden layer, having T number of neurons. Here, δ is termed *activation* or *threshold* function, which in a way quantize the output of the network. The most commonly used threshold functions are *step*, *linear*, and *tan-sigmoid* (hyperbolic tangent sigmoid):

$$Y_k = \delta \left(\sum_{j=0}^M w_{kl}^y \delta_H \left(\sum_{j=0}^M w_{ij}^H P_R \right) \right). \quad (1)$$

Fig. 3 Structure of fully connected feed-forward neural network with a single hidden layer

Selection of the number of hidden layers and the number of neurons per layer are a critical process, having a high impact on the systems stability. Most of the available ANN training algorithms utilize MSE (difference between the expected and observed responses) as their objective function:

$$\phi = \frac{1}{2} \sum_{k=1}^M (y_k - d_k)^2 = \frac{1}{2} \sum_{k=1}^M e_k^2 \quad (2)$$

where y_k is the k th output value calculated by d_k , and it represents the expected value. To the best of our knowledge, all the ANN architectures follow *backpropagation* technique to minimize their objective function—from the basic feed-forward neural network to those widely adopted architectures, such as convolutional neural network (CNN), all the training algorithms back propagate their error in the form of sensitivities from the output to input layer.

ANN architectures

Feedforward

The simplest architecture that an ANN model may follow is termed a feed-forward neural net, in which each layer is only connected to its immediate neighbors. The mapping from input through the hidden to the output layers, therefore, is achieved in a serial and linear manner.

Cascaded

Unlike the feed-forward nets, the output layer in a cascaded network is not only connected to its immediate predecessor (hidden) layer, but also has a direct connection to the input layer. This allows the network to exploit the initial weights

in addition to those provided by the predecessor to facilitate optimal convergence at the cost of added complexity.

Layer-recurrent

Unlike the feed-forward nets, the layer-recurrent nets are nonlinear, i.e., they have a feedback loop in the hidden layers with additional tap delays. The latter especially prove helpful in analyzing time-series data, where the network is supposed to have a dynamic response.

Parts (a), (b), and (c) in Fig. 4 depict feed-forward, cascaded, and layer-recurrent neural nets, respectively, where circles in (c) depict the additional tap delays.

Benchmark backpropagation algorithms

The backpropagation technique is an iterative method, which works in conjunction with gradient descent algorithm (Reed et al. 1993). While in each iteration, the network coefficients are updated, the method continues to compute gradient of the cost function accordingly. The objective of this iterative method is to minimize the cost function in terms of MSE:

$$\nabla \phi(w) = \frac{\partial \phi(w)}{\partial w_j} = 0 \quad \forall j \quad (3)$$

update rule:

$$w(k+1) = w(k) + \nabla w(k)$$

where

$$\nabla w(k) = -\alpha \frac{\partial \phi(k)}{\partial w(k)}$$

where α is the learning parameter.

Because of its broad range of contributions, several variants of backpropagation algorithm have been proposed. Although each variant has its own pros and cons, in what follows, we discuss only the ones that have proven their efficiency for the problems as such addressed in the proposed work (Haider et al. 2017).

Levenberg Marquardt framework

The Levenberg Marquardt (LM) algorithm is a pseudo-second-order training algorithm, which works in conjunction with the steepest descent method. It has been reported that this algorithm promises better stability and convergence speed (Levenberg et al. 1944).

Let us consider the output response of feedforward neural network, calculated using Eq. 1, where initial output response is given as $y_0 = r_k$. The network error is calculated using Eq. 2.

The network sensitivities are backpropagated through the network to update the learning rules (Demuth et al. 2014). Derived from the Newton algorithm and steepest descent method, the update rule for LM algorithm is defined as

$$\Delta W = (J_{we}^T J_{we} + \delta_r I)^{-1} J_{we}^T e \quad (4)$$

or the above equation can be written as

Fig. 4 ANN architectures: **a** feedforward, **b** cascaded, and **c** layer recurrent

$$\Delta x_k = -[J_{we}^T(x_k)J_{we}(x_k) + \delta_r I]^{-1} J_{we}^T(x_k)v(x_k) \quad (5)$$

where J_{we} has dimensions $(P \times Q \times R)$ and error vector is the matrix of dimensions $(P \times Q \times 1)$. The Jacobian matrix is defined using relation:

$$J_{we} = \begin{bmatrix} \frac{\partial e_{11}}{\partial w_1} & \frac{\partial e_{11}}{\partial w_2} & \cdots & \frac{\partial e_{11}}{\partial w_R} & \frac{\partial e_{11}}{\partial b_1} \\ \frac{\partial e_{12}}{\partial w_1} & \frac{\partial e_{12}}{\partial w_2} & \cdots & \frac{\partial e_{12}}{\partial w_R} & \frac{\partial e_{12}}{\partial b_1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{\partial e_{1Q}}{\partial w_1} & \frac{\partial e_{1Q}}{\partial w_2} & \cdots & \frac{\partial e_{1Q}}{\partial w_R} & \frac{\partial e_{1Q}}{\partial b_1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{\partial e_{P2}}{\partial w_1} & \frac{\partial e_{P2}}{\partial w_2} & \cdots & \frac{\partial e_{P2}}{\partial w_R} & \frac{\partial e_{P2}}{\partial b_1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{\partial e_{P2}}{\partial w_1} & \frac{\partial e_{P2}}{\partial w_2} & \cdots & \frac{\partial e_{P2}}{\partial w_R} & \frac{\partial e_{P2}}{\partial b_1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{\partial e_{PQ}}{\partial w_1} & \frac{\partial e_{PQ}}{\partial w_2} & \cdots & \frac{\partial e_{PQ}}{\partial w_R} & \frac{\partial e_{PQ}}{\partial b_1} \end{bmatrix} \quad (6)$$

where P is the number of training patterns with Q outputs, R is the number of weights and elements in error vector, and e is calculated using Eq. 2. Conventionally, Jacobian matrix J is initially calculated and later computations are performed on stored values for weights and biases updation. With fewer patterns, this method works smoothly and efficiently while with large sized patterns, calculation of Jacobian matrix faces memory limitations. This concludes LM algorithm performance degraded with larger training patterns.

(CGF) (Naqvi et al. 2016; Johansson et al. 1991; Powell 1977).

Let us consider set of input vectors r_k , which is mutually conjugate with respect to positive definite Hessian matrix H_{wb} , according to condition:

$$r_k^T H_{wb} r_k = 0. \quad (7)$$

The quadratic function is minimized by searching along the eigenvectors of the Hessian matrix H_{wb} . For the give iterative function

$$\nabla F(w) - H_{wb} r_k + \varpi \quad (8)$$

$$\nabla^2 F(w) = H_{wb}. \quad (9)$$

For the iteration $k + 1$, the change in gradient can be calculated from equation:

$$\Delta U_k = U_{k+1} - U_k = (H_{wb} r_{k+1} + \varpi) - (H_{wb} r_k + \varpi) = H_{wb} \Delta r_k \quad (10)$$

where

$$\Delta r_k = (r_{k+1} - r_k) = \delta_R^k r_k \quad (11)$$

where δ_R is a learning rate and is selected to minimize function $F(w)$ in the direction of r_k . The first search direction is arbitrary:

$$r_0 = -U_0 \quad (12)$$

where

$$U_k \equiv \nabla F(w)|_{w=w_0}. \quad (13)$$

Algorithm 1: Levenberg Marquardt backpropagation

Result: Weights & biases update for curve approximation

- 1 Estimate the network's response using eqn. 1 and its mean squared error can be computed using eqn. 2.
 - 2 Construct the Jacobian matrix using eqn. 6 and calculate sensitivities [28].
 - 3 **while** $TolLevel < \epsilon$ **do**
 - 4 Solve eqn. 5 to obtain Δx_k
 - 5 Recompute the sum of squared errors x_{k+1} using $x_k + \Delta x_k$
 - 6 **if** $x_{k+1} < x_k$ **then**
 - 7 Divide δ_r factor in eqn. 5 with user defined variable ϱ
 - 8 **end**
 - 9 **else**
 - 10 Multiply δ_r factor in Eqn. 5 with user defined variable ϱ
 - 11 **end**
 - 12 **end**
-

Conjugate gradient

The Conjugate Gradient (CG) algorithm is known for its fast convergence rate, and has been employed for solving sparse linear equations on numerous occasions. Few of its variants include Scaled CG (SCG) and Fletcher–Powell CG

The Gram–Schmidt Orthogonalization (Messouadi 1996) is used to construct r_k for each iteration, orthogonal to $\{\Delta U_0, \Delta U_1, \dots, \Delta U_{k-1}\}$ as

$$r_k = -U_k + \beta_k r_{k-1} \quad (14)$$

where β_k is scalar and given as

$$\beta_k = \frac{\Delta U_{k-1}^T U_k}{U_{k-1}^T U_{k-1}} \quad (15)$$

and δ_R can be calculated using relation:

$$\delta_R^k = \frac{-U_k^T U_k}{r_k^T H_{wb} r_k}. \quad (16)$$

large set of IV curves. We divide the set into two: (1) training and testing samples and (2) sample for cross-checking, and use the first subset to train our 90 ANN models (3 architectures $\times$ 10 configurations $\times$ 3 training algorithms), where each configuration refers to a different number of neurons in the hidden layers. Once trained, the system provides us the training time and epochs taken by

Algorithm 2: Conjugate Gradient

Result: Estimated curve by updated coefficients

```

1 Eqn. 12 initializes the search direction
2 while TolLevel <  $\epsilon$  do
3   Use eqn. 16 to find  $\delta_R^k$ ;
4   Use eqn. 11 identifies the next step;
5   Eqn. 14 and eqn. 15 indicate the next search direction and  $\beta_k$ ;
6 end
```

Bayesian regularization

The Bayesian Regularization (BR) algorithm makes use of LM algorithm (hence not so different) to search for the minima from Hessian matrix of a given function. Refer to MacKay (1992) for the detailed description.

Research methodology and simulation results

Research methodology

Figure 5 concisely presents our data collection and analysis methodology. We have used the PPMS for obtaining a

each ANN model to converge; we record these values. Following the successful training, validation, and testing phases (combined called ANN Learning), we predict the response for the second sample, and record the MSE between the predicted and measured responses. The same process is repeated for ten configurations.

Objective statement

Let $\Omega \subset \mathbb{R}^l$, where $l \rightarrow \{R \times D\}$, such that $R > D$, be a bounded experimental data. Let $\phi = \phi(i) | i \in \Omega$ be selected features set, $\phi_1(i), \dots, \phi_n(i) \subset \phi$ are n features associated with training and testing of ANN's network to predict output vector $\tilde{\phi}_{pred}$. Formally, ϕ is mapped to $\tilde{\phi}_{pred}$:

Fig. 5 Adopted research methodology

Table 1 Curves used for comparison

Geometry	Temperature (K)	Magnetic flux (Oe)
Rectangle	8.8	0
Square	8.65	41
Honeycomb	8.6	200
Kagome	8.4	2000

$\phi \rightarrow \tilde{\phi}_{pred}$. The output vector $\tilde{\phi}_{pred}$ is the predicted version of ϕ in terms of

$$\{\tilde{\phi}_{pred} \triangleq (\delta_{WB}^k, \delta_{NN}^2, \delta_R, \delta_S) \in \{-1 : 1\}, (\delta_{WB}^L, \delta_{WB}^C, \delta_{WB}^B) \in \mathbb{R}^l\}$$

where the input parameters, δ_{WB}^k is a vector of randomly initialized weights and biases, δ_{NN}^2 represents two hidden layers with different combinations of neurons, and δ_R and δ_S are the learning rate and step size, respectively. The output parameters comprised of optimized weights and biases vector from three different training algorithms of δ_{WB}^L (LM algorithm), δ_{WB}^C (CG algorithm) and δ_{WB}^B (BR algorithm). Performance parameters are selected to be

MSE, number of epochs, and training time. The cost function is selected to be MSE and calculated using Eq. 2

Simulation results

The entire purpose of this research work is to highlight the ability of the proposed approach to predict the IV curves for different values of temperature and magnetic flux, which were not available while training the ANN model. As already stated in “Physical measurement system and readings” section, we kept four IV curves (one for each geometry of antidots) isolated from the modeling data, which we should use for cross-checking of the proposed approach on unforeseen data. Table 1 presents those IV curves. Figure 6 presents a plot of the predicted values against the physically measured ones. The thicker curves, for example, for the square array (shown in the top right corner), represents larger number of data points available in the transport measurements. The reason for choosing a different number of data points for each geometry was twofold. First, we deliberately wanted to evaluate performance of each training algorithm in the absence of

Fig. 6 Predicted IV curves: rectangular (top left), square (top right), honeycomb (bottom left), and kagome (bottom right)

Fig. 7 MSE: rectangular (top left), square (top right), honeycomb (bottom left), and kagome (bottom right)

sufficiently large data sets, and thereby estimate the minimum data points needed for an acceptable MSE. Second, we wanted to showcase the applicability of our approach on various geometries with a varying number of data points in the transport measurements. It may be clearly observed that prediction for each geometry results in a negligible error.

Figures 7, 8, 9, respectively, present MSE, number of iterations to converge, and training time by each of the training algorithms. Note that the horizontal axis in each figure corresponds to different ANN models; each having a different number of neurons in the hidden layer; we call this network configuration. In essence, each plot corresponds to a different geometry of antidots, trained by three benchmark algorithms for thirty different configurations.

Considering the fact that training an ANN model is a stochastic process—immensely relying upon random numbers—it is difficult to ascertain the reason behind diversity, especially sharp peaks, in each result. Therefore, it is not possible to advocate the use of one algorithm for

all the geometries and architectures. Instead, let us comment on the obtained results in a case-by-case manner. BR algorithm outperforms the other two in terms of MSE for the square and honeycomb arrays, which had more data points than the remaining geometries: rectangular and kagome. However, this happens only at the cost of increased training time and number of iterations to converge. The increased MSE in case of the latter two geometries by BR reflects its impotence in approximating curves with a dearth of data points. For these two geometries, LM appears to be more promising, except for a few random peaks, see Fig. 7.

LM and CGF prove to be a better option if fast convergence, both in terms of number of iterations and training time, is decisive. This is evident in Figs. 8 and 9. It is interesting to note that CGF, in contrast to BR, takes a large number of iterations to converge for square and honeycomb arrays; those having large data sets, and its training time is minimal with geometries having smaller data sets. This advocates its usage in systems requiring real-time approximation, where accuracy could be slightly

Fig. 8 Epochs: rectangular (top left), square (top right), honeycomb (bottom left), and kagome (bottom right)

compromised. However, for the application presented in this work, CGF is not the best available option. On the other hand, LM stands between the other two alternative algorithms, both in terms of prediction accuracy and convergence rate. While it has better approximation accuracy for smaller data sets, it tends to converge faster for the geometries having large number of data points.

Table 2 presents the best results, in terms of minimum MSE, epochs, and training time, obtained from the prediction process for each geometry. Note that number (No.) of neurons, expressed as $[x, y]$, represents x and y neurons in the first and second layers of each neuron, respectively. The table should be interpreted as follows: for the rectangular array, the layer-recurrent architecture having eleven and five neurons in the hidden layers, once trained with LM algorithm, achieves the MSE of 3.3×10^{-7} , which is better than any other pair or architecture and algorithm. Similarly, for the same geometry, BR converges in the least number of iterations of 11 with the layer-recurrent architecture, having 18 and 10 neurons in the hidden layer, while CGF trains the cascaded network with $[5, 2]$ neurons in just 0.093 s, which is faster than all other options. It is evident that BR, if provided with a sufficiently large data set, can

outperform the rest of the algorithms in terms of MSE, whereas LM and CGF can be good options for minimum training time and epochs, even in the absence of large data sets. For the purpose of predicting IV curves in superconducting films, this work will be used as benchmark, since it points out the best pairs of architecture and algorithm for the most widely adopted assessment parameters, namely: MSE, epochs, and training time.

Conclusion

Motivated by the experience that transport measurements in superconducting films are notorious and cumbersome to obtain, a predictive model, based on artificial neural networks, has been proposed. The model takes a finite amount of data points, for each of the four geometries of antidots including rectangular, square, honeycomb, and kagome, and extrapolates the curves for a wide range of unforeseen values of temperature and magnetic flux. We have assessed three different architectures of artificial neural networks, trained by three renowned training algorithms for the purpose of predicting these current–voltage curves. Our

Fig. 9 Time: rectangular (top left), square (top right), honeycomb (bottom left), and kagome (bottom right)

Table 2 Best results in terms of MSE, epochs, and training time

Parameter	Architecture	No. of neurons	Training function	Min value
(a) Rectangular				
MSE	Layer recurrent	[11 5]	LM	3.3×10^{-7}
Epochs	Layer recurrent	[18 10]	BR	11
Training Time	Cascaded	[5 2]	CGF	0.093 s
(b) Square				
MSE	Layer recurrent	[10 12]	BR	3.8×10^{-9}
Epochs	Feedforwards	[12 6]	LM	17
Training Time	Cascaded	[5 2]	LM	3.43 s
(c) Honeycomb				
MSE	Cascaded	[12 10]	BR	7.11×10^{-9}
Epochs	Layer recurrent	[5 2]	LM	11
Training Time	Feedforward	[5 2]	LM	0.84 s
(d) Kagome				
MSE	Layer recurrent	[10 5]	LM	9.4×10^{-8}
Epochs	Layer recurrent	[15 6]	CGF	24
Training Time	Cascaded	[5 2]	CGF	0.14 s

assessment is based on mean-squared error, number of iterations to converge, and training time. Our simulations have pointed out the attributes of each architecture and algorithm, which should help all the followup works to make a choice between the available options as desired—giving this study a significant importance in the field of computational physics.

Open Access This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

References

- Baert M et al (1995) Composite flux-line lattices stabilized in superconducting films by a regular array of artificial defects. *Phys Rev Lett* 74(16):3269
- Cuppens J, Ataklti GW, Gillijns W, Van de Vondel J, Moshchalkov VV, Silhanek AV (2011) Vortex dynamics in a superconducting film with a kagom and a honeycomb pinning landscape. *J Supercond Novel Magn* 24(1):7–11
- Cybenko G (1989) Approximation by superpositions of a sigmoidal function. *Math Control Signals Syst (MCSS)* 2(4):303–314
- de Lara DP, Alija A, Gonzalez EM, Velez M, Martin JI, Vicent JL (2010) Vortex ratchet reversal at fractional matching fields in kagomlike array with symmetric pinning centers. *Phys Rev B* 82(17):174503
- Demuth HB, Beale MH, De Jess O, Hagan MT (2014) Neural network design. Martin Hagan, Boston
- Elminir HK, Azzam YA, Younes FI (2007) Prediction of hourly and daily diffuse fraction using neural network, as compared to linear regression models. *Energy* 32(8):1513–1523
- Ghanbari A, Naghavi A, Ghaderi SF, Sabaghian M (2009) Artificial Neural Networks and regression approaches comparison for forecasting Iran's annual electricity load. In International conference on power engineering, energy and electrical drives, 2009. POWERENG'09, pp. 675–679
- Gucul U, van Gerven MAJ (2015) Deep neural networks reveal a gradient in the complexity of neural representations across the ventral stream. *J Neurosci* 35(27):10005–10014
- Guojin C, Miaofen Z, Honghao Y, Yan L (2007) Application of neural networks in image definition recognition. In: IEEE International conference on signal processing and communications, 2007. ICSPC 2007, pp. 1207–1210
- Haider SA, Naqvi SR, Akram T, Kamran M (2017) Prediction of critical currents for a diluted square lattice using Artificial Neural Networks. *Appl Sci* 7(3):238
- He SK, Zhang WJ, Liu HF, Xue GM, Li BH, Xiao H, Wen ZC et al (2012) Wire network behavior in superconducting Nb films with diluted triangular arrays of holes. *J Phys Condens Matter* 24(15):155702
- Hornik K (1991) Approximation capabilities of multilayer feedforward networks. *Neural Netw* 4(2):251–257
- Hornik K (1993) Some new results on neural network approximation. *Neural Netw* 6(8):1069–1072
- Hornik K, Stinchcombe M, White H (1989) Multilayer feedforward networks are universal approximators. *Neural Netw* 2(5):359–366
- Jaccard Y, Martin JI, Cyrille M-C, Vlez M, Vicent JL, Schuller IK (1998) Magnetic pinning of the vortex lattice by arrays of submicrometric dots. *Phys Rev B* 58(13):8232
- Johansson EM, Dowla FU, Goodman DM (1991) Backpropagation learning for multilayer feed-forward neural networks using the conjugate gradient method. *Int J Neural Syst* 2(04):291–301
- Kamran M, Naqvi SR, Kiani F, Basit A, Wazir Z, He SK, Zhao SP, Qiu XG (2015) Absence of reconfiguration for extreme periods of rectangular array of holes. *J Supercond Novel Magn* 28(11):3311–3315
- Kamran M, Haider SA, Akram T, Naqvi SR, He SK (2016) Prediction of IV curves for a superconducting thin film using artificial neural networks. *Superlattices Microstruct* 95:88–94
- Latimer ML, Berdiyorov GR, Xiao ZL, Kwok WK, Peeters FM (2012) Vortex interaction enhanced saturation number and caging effect in a superconducting film with a honeycomb array of nanoscale holes. *Phys Rev B* 85(1):012505
- Levenberg K (1944) A method for the solution of certain non-linear problems in least squares. *Q Appl Math* 2(2):164–168
- MacKay DJC (1992) A practical Bayesian framework for backpropagation networks. *Neural Comput* 4(3):448–472
- Marquardt DW (1963) An algorithm for least-squares estimation of nonlinear parameters. *J Soc Ind Appl Math* 11(2):431–441
- Martin JI, Velez M, Nogues J, Schuller IK (1997) Flux pinning in a superconductor by an array of submicrometer magnetic dots. *Phys Rev Lett* 79(10):1929
- Martin JI, Vlez M, Hoffmann A, Schuller IK, Vicent JL (1999) Artificially induced reconfiguration of the vortex lattice by arrays of magnetic dots. *Phys Rev Lett* 83(5):1022
- Meireles MRG, Almeida PEM, Simes MG (2003) A comprehensive review for industrial applicability of artificial neural networks. *IEEE Trans Ind Electron* 50(3):585–601
- Messaoudi A (1996) Recursive interpolation algorithm: a formalism for solving systems of linear equations II Iterative methods. *J Comput Appl Math* 76(1–2):31–53
- Mostako ATT, Khare A (2012) Effect of target substrate distance onto the nanostructured rhodium thin films via PLD technique. *Appl Nanosci* 2(3):189–193
- Naqvi SR, Akram T, Haider SA, Kamran M (2016) Artificial neural networks based dynamic priority arbitration for asynchronous flow control. *Neural Comput Appl* 1–11. <https://doi.org/10.1007/s00521-016-2571-6>
- Odagawa A, Sakai M, Adachi H, Setsune K, Hirao T, Yoshida K (1997) Observation of intrinsic Josephson junction properties on (Bi, Pb) SrCaCuO thin films. *Jpn J Appl Phys* 36(1A):L21
- Powell MJD (1977) Restart procedures for the conjugate gradient method. *Math Program* 12(1):241–254
- Quan G-z, Pan J, Wang X (2016) Prediction of the hot compressive deformation behavior for superalloy nimonic 80A by BP-ANN model. *Appl Sci* 6(3):66
- Reed R (1993) Pruning algorithms—a survey. *IEEE Trans Neural Netw* 4(5):740–747
- Reyen ME, Grkan P (2008) Comparison of artificial neural network and linear regression models for prediction of ring spun yarn properties. I. Prediction of yarn tensile properties. *Fibers Polym* 9(1):87–91
- Setti SG, Rao RN (2014) Artificial neural network approach for prediction of stress-strain curve of near titanium alloy. *Rare Met* 33(3):249–257
- Shahid MU, Deen KM, Ahmad A, Akram MA, Aslam M, Akhtar W (2016) Formation of Al-doped ZnO thin films on glass by sol-gel process and characterization. *Appl Nanosci* 6(2):235–241
- Villegas JE, Savelev S, Nori F, Gonzalez EM, Anguita JV, Garcia R, Vicent JL (2003) A superconducting reversible rectifier that

controls the motion of magnetic flux quanta. *Science* 302(5648):1188–1191

Zhao M, Li Z, He W (2016) Classifying four carbon fiber fabrics via machine learning: a comparative study using ANNs and SVM. *Appl Sci* 6(8):209

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.