Neural Computing and Applications (2021) 33:1923–1933
https://doi.org/10.1007/s00521-020-05035-x

ORIGINAL ARTICLE

An efficient optimization approach for designing machine learning models based on genetic algorithm

Khader M. Hamdia[1] · Xiaoying Zhuang[2,3] · Timon Rabczuk[4]

Received: 5 December 2019 / Accepted: 12 May 2020 / Published online: 20 June 2020

Abstract

Machine learning (ML) methods have shown powerful performance in different application. Nonetheless, designing ML models remains a challenge and requires further research as most procedures adopt a trial and error strategy. In this study, we present a methodology to optimize the architecture and the feature configurations of ML models considering a supervised learning process. The proposed approach employs genetic algorithm (GA)-based integer-valued optimization for two ML models, namely deep neural networks (DNN) and adaptive neuro-fuzzy inference system (ANFIS). The selected variables in the DNN optimization problems are the number of hidden layers, their number of neurons and their activation function, while the type and the number of membership functions are the design variables in the ANFIS optimization problem. The mean squared error (MSE) between the predictions and the target outputs is minimized as the optimization fitness function. The proposed scheme is validated through a case study of computational material design. We apply the method to predict the fracture energy of polymer/nanoparticles composites (PNCs) with a database gathered from the literature. The optimized DNN model shows superior prediction accuracy compared to the classical one-hidden layer network. Also, it outperforms ANFIS with significantly lower number of generations in GA. The proposed method can be easily extended to optimize similar architecture properties of ML models in various complex systems.

Keywords Machine learning · Deep neural networks · Optimization · Genetic algorithm · Polymer nanocomposites · Fracture energy.

List of symbols

ML	Machine learning
ANN	Artificial neural networks
DNN	Deep neural networks
ANFIS	Adaptive neuro-fuzzy inference system
GA	Genetic algorithm
MSE	Mean squared error
R^2	Coefficient of determination
PNCs	Polymer/nanoparticles composites
$\mathcal{G}(\cdot)$	The activation function
f^l	The output of a hidden layer
$\boldsymbol{w}$	The connecting weights
k	Learning iteration
$\mathbf{e}$	The error vector
$\mathbf{J}$	The Jacobian matrix for the first derivatives of the network errors with respect to the weights
μ	The learning rate
$\mathbf{I}$	The identity matrix
f_i	Crisp function of ANFIS
$\mu_i(x_i)$	Membership grade of the inputs, x_i
$\boldsymbol{y}$	The set of n integer state variables
$p(\boldsymbol{y})$	The fitness function
V_f	Volume fraction of nanofiller
d_n	The diameter of the nanoparticles
G_{Im}	The fracture energy
E_m	Young's modulus
σ_{ym}	The yield strength
N	The number of datasets
t_i	The target output
O_i	The predicted output by ML model

✉ Xiaoying Zhuang
xiaoying.zhuang@tdtu.edu.vn

[1] Chair of Computational Science and Simulation Technology, Department of Mathematics and Physics, Leibniz Universität Hannover, Appelstr. 11, 30167 Hannover, Germany

[2] Division of Computational Mechanics, Ton Duc Thang University, Ho Chi Minh City, Vietnam

[3] Faculty of Civil Engineering, Ton Duc Thang University, Ho Chi Minh City, Vietnam

[4] Institute of Structural Mechanics, Bauhaus-Universität Weimar, Marienstr. 15, 99423 Weimar, Germany

N_f The total number of the model evaluations
N_w The number of connecting weights between layers

1 Introduction

Machine learning (ML) methods have been extensively used for simulating material design in various applications recently thanks to the considerable advancements in computing power. The high advantage of ML is to represent the actual behavior with much less cost and time. These tools are based on computational intelligence through correlating the input parameters to the output/s of interest by means of mathematics and statistical methods. ML modeling is reasonably accurate and able to capture and identify the nonlinearity in the very complex physical systems by developing a black box model without the need to mathematical models. Thus, it has become a viable complement and even an alternative to the physically based model [1–3]. Artificial neural network (ANN) is a widely common ML method that has the ability to learn the pattern rapidly. Disregarding the nature of the problem understudy, all the influencing factors can be taken into account considering their complicated joint effect. The general structure of ANN is composed of parallel layers connected by weights and biases to form the network. Feed-forward neural networks are used to construct an approximate function for the relationship starting from the input layer toward the output layer and passing through a hidden layer. The weights and the biases can be learned making use of a predefined training process [4, 5]. Different from the classical shallow ANN, the deep neural network (DNN) is formed by multiple processing hidden layers (more than two) providing a higher learning representation [6]. Moreover, the adaptive neuro-fuzzy inference system (ANFIS) presents a combination of neural network and a fuzzy system that deals with reasoning. Using these artificial intelligence approaches, the behavior of the given problems can be captured effectively and, consequently, the future response can be predicted implicitly with much less effort.

The structure and the related configurations are essential modeling factors in building the ML model. Different results are obtained when the architecture and the feature configurations are changed. The numbers of input and output variables define the number of neurons in the input and output layers. In the meantime, there is still no general rule for setting the dimension of the hidden layers and the number of neurons in each layer. It is difficult to find the optimal set of the possible structures and parameters. A trial and error is a very common adopted procedure in which the tedious iterative process is inevitable. Though, reliable and powerful optimization methods have been effectively used in identifying the optimal model from several trained models. Much of the focus has been given on mainly optimizing the parameters that can be derived through the training process. The optimal model was selected by finding the optimal connecting weights and was approximated by reducing the training error between the predictions and corresponding targets [7–9]. In ANFIS models, most of the optimization techniques were utilized for defining the membership functions and the corresponding fuzzy rules to increase the accuracy of standalone ANFIS [10, 11]. When it comes to optimal architecture, it is seen that limited analyses were investigated. An efficient configuration of ML models can be obtained by optimizing hyperparameters whose magnitude is to be set before the learning process begins. This is a complicated optimization problem as it contains a large number of correlated design variables and nonlinear objective function. Therefore, a method for developing and optimizing ML models to obtain the best model configuration is needed.

This paper presents a robust methodology for finding the optimal architecture and features of the DNN and ANFIS models. Supervised learning is considered where the data points include a target output to be predicted from a given set of input variables. Genetic algorithm (GA)-based integer-valued optimization is employed to find the best ML model configuration through minimizing a fitness function of the mean squared error (MSE) between the predicted and the target values. The hyperparameters are restricted to be integers. For DNN, the optimization variables include: the number of hidden layers, the number of neurons in each one, and the type of the activation function, whereas for ANFIS, they are the type and the number of membership functions In addition, the performance of the addressed ML models is also evaluated and compared by calculating the corresponding coefficient of determination (R^2) and the probability distribution of the relative error.

We apply the proposed methodology to the computational design of materials in order to validate the method and compare it with the classical technique. The focus of interest in this paper is the prediction of the fracture energy of polymer/nanoparticles composites (PNCs) based on a set of experimental measurements gathered from the literature. This is a challenging task considering the complex and nonlinear nature of toughening mechanism of PNCs which depends on diverse uncertain factors. Up to date, there are only a few contributions on ML to investigate the behavior of PNCs [12–14]. In a previous study, we presented unoptimized ML models for PNCs that were constructed

using the concept of trial and error [15]. The developed ANN and ANFIS showed a considerable superior performance compared to the results obtained by the analytical and linear regression models. Yet, the search for a better model never stops. To the best knowledge of the authors, the optimal design of the architecture and the hyperparameters of the ML model in the area of PNCs material design remains almost unexplored.

The rest of the paper is organized as follows: firstly, we describe the machine learning and the optimization methods with their essential mathematical background in Sect. 2. Then, Sect. 3 addresses the model problem including details of the dataset. Subsequently, we briefly describe the application and performance's analysis of the proposed method in Sect. 4. Finally, Sect. 5 summarizes the key results and provides direction for future works.

2 Material and methods

There exist several ML approaches documented in the literature such as decision trees, random forest, support vector, Bayesian networks, extreme learning machine, evolutionary computing, ANN, and ANFIS, among others. An extensive review on the methods can be found in [16–18]. Nevertheless, for the purpose of finding the optimal configurations of ML model, ANN and ANFIS are investigated in this study due to their flexible and adjustable architecture. We have not performed a conclusive or exhaustive analysis to determine whether and how they are better, as our major concern to have relatively simple and optimized model that could be easily applied. Hereafter, a short description and the mathematical formulations involved in the development of the proposed method are elucidated.

2.1 Artificial neural networks

Artificial neural network (ANN) is a highly parallel system that mimics the function of the biological brain. It is designed to model the relation among the input and output parameters through a training process. The typical ANN model contains several inter-connected processing units called neurons or nodes grouped together into layers. The neighboring layers are connected by weights forming a large network. The network learns by analyzing multiple datasets and adjusting the connection weights [19, 20]. In the course of this study, we apply the multilayer feedforward networks to predict the fracture energy of the PNCs. The optimal architecture of the deep neural network (DNN), a network composed of two or more hidden layers, is examined. The architecture of the network and the proposed DNN model for predicting the fracture energy of PNCs is shown in Fig. 1.

Inputs from previous layers are linked to each neuron by the corresponding weights and bias by which the neuron receives data and consequently proceeds it to the next layer. The weighted sums are passed through an activation function, $\mathcal{G}(\cdot)$, to determine the neuron output. It takes inputs from previous layer to produce a scalar output. The output is computed layer by layer as in Eq. (1).

$$f^l = \mathcal{G}(\boldsymbol{w}^l \cdot f^{l-1} + b^l) \tag{1}$$

where f^{l-1} is the output from the preceding layer $(l-1)$, and $\boldsymbol{w}$ and b are the connecting weights and bias, respectively. Finally, the signal from the neurons of the last hidden layer is passed to the output layer with linear activation [21, 22]. The network learns iteratively from several datasets in supervised learning process. The predicted outputs are compared with the target output, and accordingly, new iteration is proceeded to minimize the mean squared error (MSE). Levenberg–Marquardt algorithm is used for training where the weights and bias are updated via the error back-propagation (BP). It is a combination of gradient descent forms and Newton method. After each learning iteration, k, the error vector ($\mathbf{e}$) is computed and the weights are updated. The Jacobian matrix ($\mathbf{J}$) includes the first derivatives of the network errors with respect to the weights. During the training in the standard gradient descent, as the error converges to a minimum value, the gradient will become very small and the weights are updated very slightly. Contrary, the training by Levenberg–Marquardt method can be much faster [23, 24]. The modification applied to update the weights is given by Eq. (2).

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k - [\mathbf{J}^T\mathbf{J} + \mu\mathbf{I}]^{-1}\mathbf{J}^T\mathbf{e} \tag{2}$$

where μ is the learning rate that governs the step size and $\mathbf{I}$ is the identity matrix.

In constructing the DNN models, we divide the data into two groups: training and testing datasets. The training dataset is used to build the network and approximate the

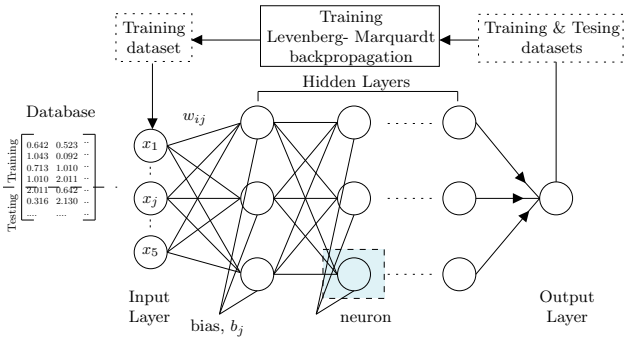

Fig. 1 The general architecture of multilayer feedforward networks for the model problem

connecting weights, while testing dataset is used to validate the models against unseen data (cross-validation) and to prevent possible overfitting. Doing so, the learning iterations continues until the stopping criterion is met; (1) the MSE in the testing set increases to a maximum number of successive iterations of 20, and (2) the gradient of the error has a minimum value of 10^{-7}.

2.2 Adaptive neuro-fuzzy inference system

The concept of fuzzy logic can be introduced through a fuzzy inference system (FIS) to map the input/output relationship. The process starts with defining the membership functions of fuzzy sets (fuzzification) statistically making use of the available dataset, comes through creating the rules and merging all the fuzzy rules by a proper fuzzy inference and ends finally by defuzzification into crisp output values [25]. The adaptive neuro-fuzzy inference system (ANFIS) benefits from the merits of fuzzy logic and the neural network [26]. It is based on using fuzzy rules for adaptation of a set of model parameters and the neural network for training and updating these parameters. Takagi–Sugeno fuzzy model is one of the ANFIS approaches characterized by linear or constant terms in the consequent part of the if-then rules [27].

The learning method of the ANFIS is similar to the common feedforward neural networks with defined network representation. It is comprised of nodes with specific functions collected in main five layers. For each layer, the output signals are processed by the node functions as displayed in Fig. 2. In the first layer, the nodes evaluate the fuzzy membership grade of the inputs, $\mu_i(x_i)$, by dividing the domain of each one into a number of fuzzy subsets. The membership function can be of increasing, decreasing or approximation type [25]. The nodes in the second layer multiply the incoming signals from Layer 1 to calculate the weights of the rules firing strength, w_i, and send the product out to the next layer. Then, the normalized firing strength of the fuzzy rules is approximated at the nodes of Layer 3, where the output in each node is calculated as the ratio of the firing weight to the sum of all firing weights. In the fourth layer, the outcomes from the preceding layer are multiplied by a crisp function (f_i) that specifies the membership function of the output. In this way, the defuzzification of the fuzzy rules is achieved for the overall weighted output. Finally, the overall output is computed as the summation of all incoming signals in Layer 5.

2.3 Genetic algorithm

For the purpose of constructing the ML models of the highest performance, the architecture and features are to be optimized in this work. The state variables represent the number of hidden layers, the number neurons in each, and the type of the activation function for the DNN, while they are the type and the number of membership functions in ANFIS. Such discrete nature of the variables makes the optimization a non-convex problem. The classical method for solving these problems is based on the branch and bound algorithm. It starts by finding the optimal solution of the variables without the integer constraints. Then, the branches of this solution are explored creating two new subproblems. The branch for each variable is checked against upper and lower estimated bounds on the optimal solution. The subproblems are solved for the new constrained and the process of branching is repeated until obtaining a solution that satisfies all the integer constraints [28]. Alternatively, heuristics methods such as genetic and evolutionary algorithms are faster and more efficient to approximate the solution of computationally expensive problems. It has been applied in solving numerical problems and prediction [29, 30]. The heuristics methods search within the domain for integer feasible solutions. Starting from randomly generated candidate, a new generation with modified objective values is extracted. The procedure is continued to derive sufficiently good solution. Like the neural networks, a genetic algorithm (GA) is biologically inspired heuristic approach based on the evolutionary process representing an optimization procedure in a binary search space. It seeks to find the values of the decision state variables that optimize an objective function. The concept of GA was presented by Holland and his collaborators [31]. In this study, GA-based integer-valued optimization is employed [32]. The mathematical formulation of the problem is defined in Eq. (3).

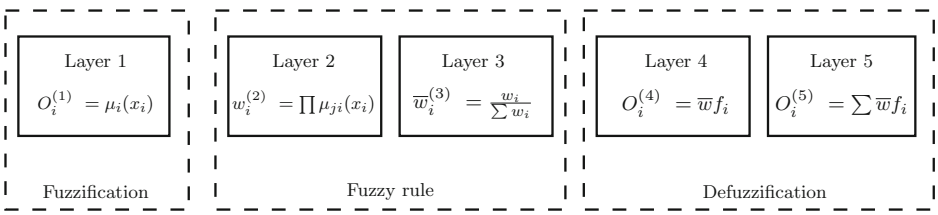

Fig. 2 Schematic representation for the layers of ANFIS

$$\begin{aligned} &\underset{\boldsymbol{y}}{\text{Minimize}}\, p\,(y) \\ &\text{Subject to}\, g(\boldsymbol{y}) = 0 \\ &\qquad h(\boldsymbol{y}) \le 0 \\ &\qquad y^l \le y_i \le y^u : integer, i = 1, \ldots, n \\ &\qquad \boldsymbol{y} = [y_1, y_2, \ldots, y_n]^T \end{aligned} \tag{3}$$

In the above expression, $\boldsymbol{y}$ is the set of n integer variables with lower and upper limits of y^l and y^u; $p(\boldsymbol{y})$ is the fitness function; and $g(\boldsymbol{y})$ and $h(\boldsymbol{y})$ are the equality and inequality constraints. The objective (fitness) function is a nonlinear function representing the MSE for the predictions of the ML models with respect to the experimental data. The flowchart of the optimization steps is shown in Fig. 3. The genetic learning starts with creation of an initial population consisting of randomly generated rules. Each rule can be represented by a string of bits. The random population is the state variables that define the architecture of the ML. Then, a new population is formed consisting the fittest rules [33]. At each step, GA uses three main types of rules to create the next generation from the current population: crossover, mutation, and selection. In the crossover, the genetic information of two parents is combined from the selection operator to explore the design space, while the genetic information is changed with mutation probability to introduce diversity and to prevent local optimal solution. Meanwhile, a selection operator is maintained to choose fit individuals for the reproduction operators. In this way, the derivatives of the objective function are not required making GA favorable choice for the nonlinear and discontinuous optimization problems. The obtained solution is evaluated by the objective optimization function of the MSE. The corresponding value of the objective (known as the fitness) measures the performance of the chosen individuals compared to the other whole population. The process of generating new populations continues as long as the termination check has not been met with the condition that each rule in the population satisfies a prespecified fitness threshold.

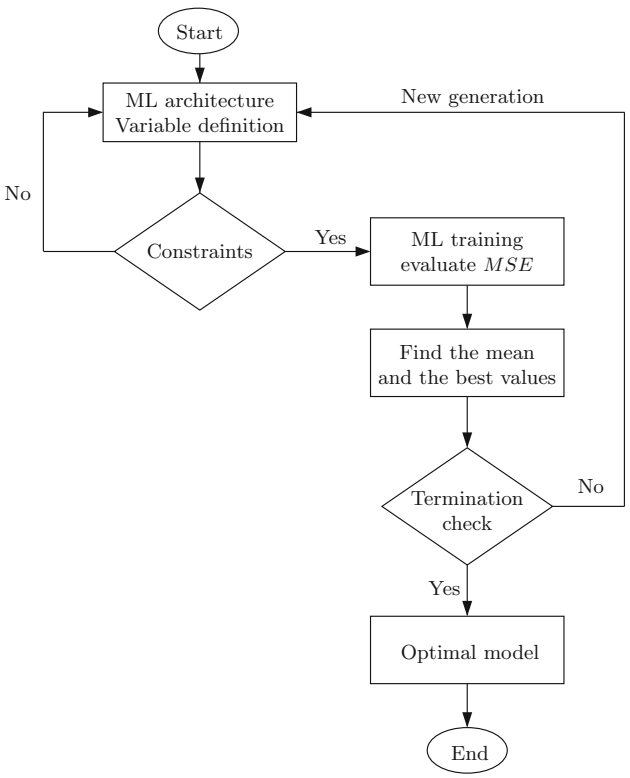

Fig. 3 Flowchart of the methodology for ML optimization using integer based GA

3 Model problem

Epoxy polymer is well known to be a brittle material. It has a poor fracture toughness and a poor resistance to crack initiation and propagation. Several diverse second-phase materials have been added to the polymer matrix in order to improve the fracture properties without sacrificing other important thermo-physical properties. Structural characterization can be enhanced for the purpose of environmental applications [34, 35]. Inorganic additives of fillers with the size of nanometer are effectively applied because of their high surface to volume ratio. In this regard, polymer nanocomposites have offered exceptional improvements even at lower filler contents. The shape of the nanofillers can be of spherical particles with a radius of 10–80 nm [36–38].

Studying the fracture energy enhancement due to the addition of rigid nanofillers is a highly challenging task. It depends on different factors which highly affect the toughening mechanism such as the volume fraction, the curing conditions, the mechanical properties of the two phases, the agglomeration, and distribution of the fillers. Several numerical and analytical models have been proposed to model the fracture and crack propagation of PNCs, see, for example, the contributions in [39–45] and the references therein. Besides these approximation models, the fracture energy of PNCs has been directly extracted from experiments [46–49].

In this study, we introduce data-driven models as a promising alternative to the 'classical' computational approaches. To establish the database for the purpose of developing the ML models, five parameters are defined. Two of them represent the geometrical properties of the rigid nanofillers, i.e., the volume fraction (V_f), and the diameter of the particle (d_n). The remaining three define the material properties of the epoxy polymers: the fracture energy (G_{Im}), the Young's modulus (E_m), and the yield

strength (σ_{ym}). These parameters are well known to be the governing factors in the toughening mechanism of PNCs according to Huang and Kinloch [39], Williams [40], and Quaresimin et al. [41]. In a previous study, we have employed ANN and ANFIS to predict the fracture energy by establishing a database collected from the literature [15]. The dataset consists of 115 experimental measurements. Various ranges are collected to ensure the development of a robust model that can be applied to a wide range of the PNCs. Table 1 lists the range of the variation of the input–output parameters.

Of the total datasets, 85 samples are assigned for training the ML models. The remaining is set as testing data to compute the validation error. It is also utilized to prevent overfitting that results in good performance in the training set and poor predictions in the overall data. Instead of random division, the data are divided based on the condition that the training and the testing datasets have identical statistical distributions.

An optimal machine learning model will be constructed to predict the fracture energy of PNCs as a function of the selected five inputs. The modeling relation is defined as

$$G_{Ic} = f(\boldsymbol{x}), \boldsymbol{x} = \{V_f, d_n, G_{Im}, E_m, \sigma_{ym}\} \quad (4)$$

where G_{Ic} is the fracture energy of the PNCS and $f()$ is the developed mapping function. An integer-valued-based GA optimization is applied to find the optimal architecture in DNN and ANFIS. In optimization scheme, various arrangements of neurons in the hidden layers using predefined multilayer neural networks of DNN are sought. In each layer, the number of neurons and the activation function are optimized in order to evaluate the fitness function. Likewise, the optimal type and number of membership functions in ANFIS are obtained.

Measuring the performance of the models generally depends on the residual of the differences between the predictions and the target values. Ordinary, absolute, or relative absolute sum of residuals is affected by the direction of over- or under-prediction. Contrary, the indicators which consider the variance are used to eliminate this effect. In our analysis, the predictions of the developed ML models are evaluated by two performance evaluation indices: the mean squared error (MSE) and the coefficient of determination (R^2) in which the variance of the estimator and its squared bias are taken into account. MSE and R^2 can be calculated, respectively, using Eqs. (5) and (6).

$$\text{MSE} = \frac{1}{N}\Sigma(t_i - O_i)^2 \quad (5)$$

$$R^2 = 1 - \frac{\Sigma(t_i - O_i)^2}{\Sigma(t_i - \bar{t_i})^2} \quad (6)$$

where N is the number of datasets, t_i refers to the actual observation from the experiments, and O_i is the predicted fracture energy by the addressed ML model. From Eq. (5), *MSE* is the variance of the residuals that corresponds to the error's discrepancy between t_i and O_i. It measures the absolute fit of the model to identify the undesirable large differences. Differently, R^2 evaluates the relative measure of the fit which quantifies the variance of the residuals divided by the total sum of squares of error with respect to the mean. Besides, the probability distribution of the relative error is employed to measure the performance of the optimal models.

Table 1 The lower and the upper limits of the dataset

Parameter	Unit	[Lower, upper] limits
Inputs		
V_f	%	[0.5,30]
d_n	nm	[12,170]
G_{Im}	J/m^2	[46.5,606]
E_m	GPa	[2.41,3.53]
σ_{ym}	MPa	[57.1,111.0]
Output		
G_{Ic}	J/m^2	[58.3,2156.6]

4 Results and discussion

The hyperparameters defining the optimal architecture of ML models are sought for the model problem of predicting the fracture energy of the PNCs (G_{Ic}). Several network architectures are examined including the shallow ANN of one hidden layer and DNN of more than two hidden layers (from 2 to 6) in order to evaluate and select the most appropriate structure. We examine also the optimal type of the activation function out of four functions: tan-sigmoid, log-sigmoid, Elliot-sigmoid, and radial basis. Apparently, these are nonlinear functions. The derivative of the linear base (e.g., triangular or pure linear) is a constant and cannot go back and modify the weights to provide a better prediction. Hence, the linear functions are not possible to use back-propagation for training the DNN models. An integer-valued-based GA is applied for this purpose with MSE minimized as the objective function. Therefore, a population size of 100 is set for the first generation and 50 for the followings. Figure 4 shows the convergence rate to obtain the minimum MSE against the number of generations in the different DNN architectures. It can be noted that *Str*.-1 has stationary trend and cannot be improved with further generations; *Str*.-6 requires 75 (= 3800 function evaluations) generations to get the lowest fitness of *MSE* = 1503.3 among the different DNNs. In the second order comes *Str*.-3 surpassing *Str*.-4 and *Str*.-5.

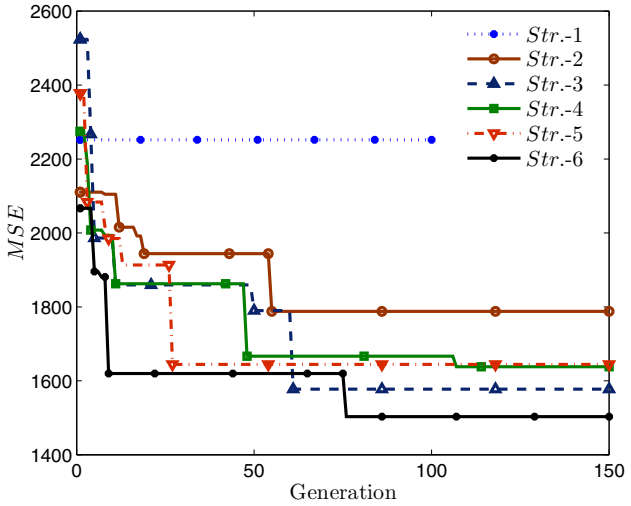

Fig. 4 Convergence curve for the objective function against the number of generations using different DNN structures

The optimal architectures of the created models are summarized in Table 2. Each row contains the optimal number of neurons in each hidden layer of the k-layered networks and their activation functions, $\mathcal{G}(\cdot)$, accompanied by the corresponding prediction accuracy measures (MSE and R^2). The last two columns include the total number of the model evaluations (N_f) used to get the optimal architecture, and the number of connecting weights (N_w) between layers. Except of the shallow network, the log-sigmoid is found to be the optimal activation function. This may be explained by its smooth gradient that prevent the jumps in the output values. Moreover, the results show that the MSE decreases steadily with the increase of hidden layers from *Str*.-1 until *Str*.-3. Afterward, it becomes slightly stabilized before it reaches the minimum at *Str*.-6. With the increase of the hidden layers, the number of connecting weights grows significantly (excluding *Str*.-4). In the comparison between *Str*.-3 and *Str*.-6, the latter, on the one hand, provides higher accuracy. On the other hand, however, it has around fourfolded number of connecting weights. Interestingly, the more the deeper of the hidden layers is, the higher the required computation burden for each function evaluation.

The network performance for the training and testing datasets in the optimal structures is shown in Fig. 5. The training error is decreasing sharply in the first ten iterations before it reaches a roughly stable convergence. The training could be continued with negligible improvement, but the error in the testing set diverges after specific number of learning iterations. The training is allowed 20 more iterations since the last time it decreased in the testing set. Hence, in order to prevent overfitting, the connecting weights are selected at the lowest MSE in the testing set. Notably, *Str*.-3 is the fastest to reach the best results.

Similarly, integer-valued-based GA optimization is applied to find the optimal hyperparameters in ANFIS model. In particular, we optimize the number and the type of the membership functions required for the fuzzification of the full space in the input parameters. The number of the rules equals the product of a sequence of the numbers of membership functions associated with the each of the five inputs. As the numbers of membership functions increase, the number of the generated rules considerably increases, and the computation time becomes unaffordable. Seven different forms of approximating membership functions are examined: Bell-shaped, Gaussian, Gaussian combination, triangular, trapezoidal, Π-shaped, and difference between two sigmoidal functions. An integer values from 1 to 7 are assigned to represent each type. Considering the number of membership functions with respect to the input parameters, the rules are generated using grid partitioning to generate a full range of the rules. The convergence of the MSE is depicted in Fig. 6. After 222 generations, the MSE converges to its minimum value of 2008.8. This convergence required 11,150 function evaluations. The computation

Table 2 The optimized architecture of DNN for predicting the fracture energy of PNCs

	N_l	Number of neurons in each hidden layer	Activation function, $\mathcal{G}(\cdot)$	MSE	R^2	N_f	N_w
Str.-1	1	[20]	Tan-sigmoid	2252.0	0.984	100	140
Str.-2	2	[34, 12]	Log-sigmoid	1787.7	0.987	1250	439
Str.-3	3	[46, 18, 25]	Log-sigmoid	1577.6	0.989	3100	1623
Str.-4	4	[33, 16, 11, 39]	Log-sigmoid	1638.2	0.988	5400	1437
Str.-5	5	[37, 38, 29, 20, 25]	Log-sigmoid	1644.1	0.988	1400	3948
Str.-6	6	[49, 37, 31, 40, 36, 12]	Log-sigmoid	1503.0	0.990	3850	6535

N_l the number of hidden layers,

N_f the total number of the objective function evaluations used to get the optimal architecture by using GA, and

N_w the number of connecting weights between layers

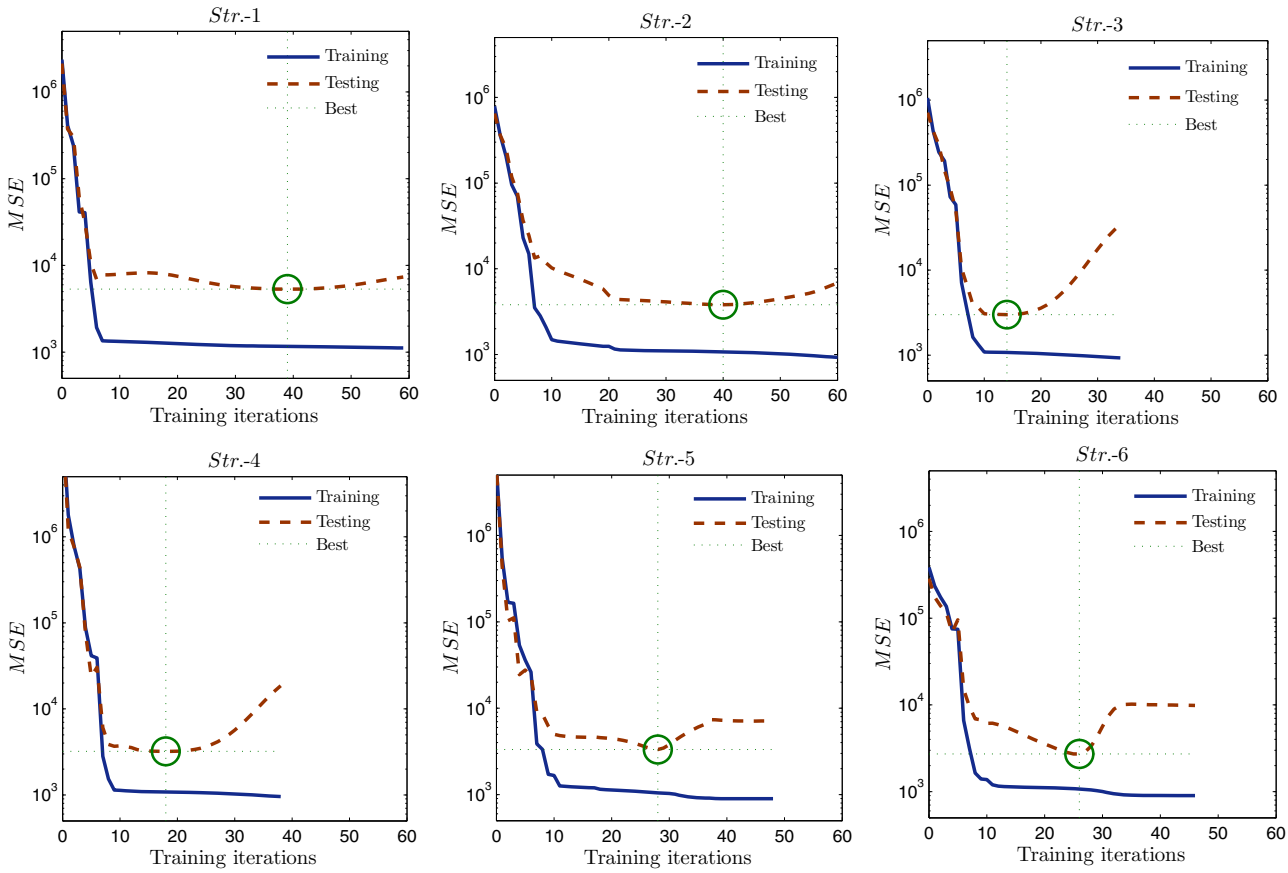

Fig. 5 Comparative convergence analyses of MSE in the training and testing datasets along the learning iterations for the optimal architectures of DNN models

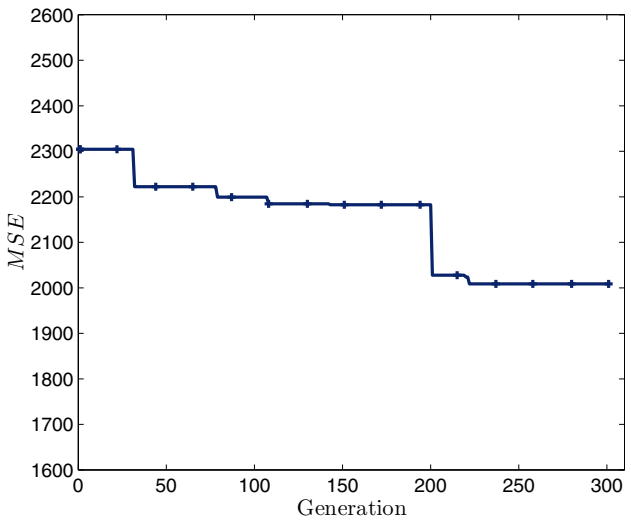

Fig. 6 Convergence curve for the objective function against the number of generations using ANFIS

cost of GA in optimizing ANFIS is significantly higher with lower predictive accuracy compared to DNN. Details on the developed optimal ANFIS are listed in Table 3. The best performance is obtained also after ten training iterations. Additional training iterations lead to an overfitting error in the testing dataset, see Fig. 7. In [15], ANFIS was found to outperform a trial selected classical one-layer ANN model. However, an optimized multiple hidden layers configuration of DNN shows better prediction accuracy.

We also employ a graphical analysis of the relative error to evaluate the performance of the optimized DNN and ANFIS models. The percentage relative error is calculated as the relative variation of the predicted value from the corresponding experimental data. The prediction inaccuracies of the optimal DNN (*Str*.-3 and *Str*.-6) and ANFIS are represented by the histograms in Fig. 8. Obviously, most of the data points have a relative error value close to zero indicating a robust predictive capability of both methods. The scatter of the distribution on either side of the equality is very similar. The probability of full matching the experimental values in the predictions by DNN is higher. The histograms show that *Str*.-6, *Str*.-3 and ANFIS can predict nearly 86%, 87%, and 76%, respectively, of the data with an absolute relative error of 10% or less.

Table 3 The optimized ANFIS for predicting the fracture energy of PNCs

Membership functions	Input-parameters					Performance indices	
	V_f	d_n	G_{Im}	E_m	σ_{ym}	MSE	R^2
Number	3	4	3	2	5		
Type	Gaussian	Triangular	Trapezoidal	Bell-shaped	Triangular	2008.8	0.986

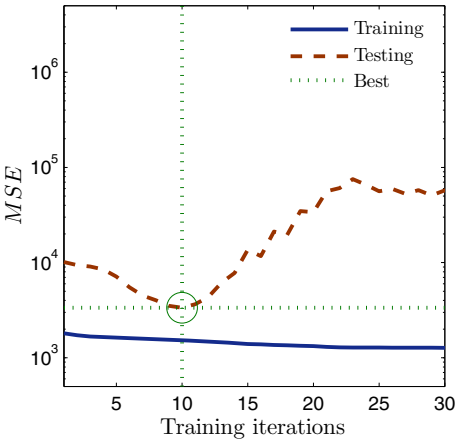

Fig. 7 Training evolution for the optimal ANFIS model

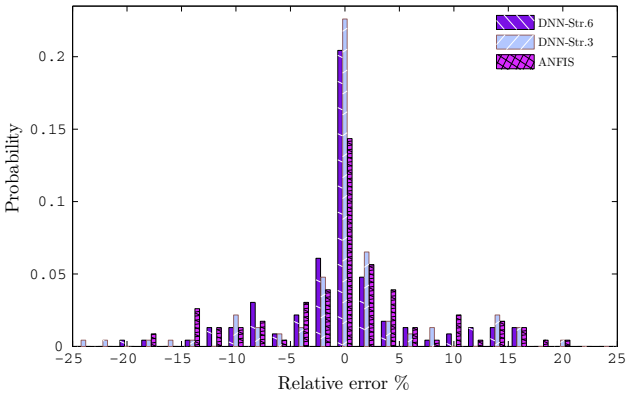

Fig. 8 Probability histograms of the relative error between predicted and experimental values

5 Conclusions

A robust optimization approach for machine learning modeling was presented in this paper based on genetic algorithm. An integer-valued optimization was adopted. The approach investigated the architecture and the feature configurations of two machine learning models: deep neural networks (DNN) and adaptive neuro-fuzzy inference system (ANFIS). The set of optimized variables were the number of neurons in the hidden layers using predefined multilayer neural networks and the type of the activation function in the former, but the type and the number of membership functions in the latter. In solving the optimization problem, the mean squared error (MSE) was assigned to be the fitness function. We elucidated the proposed method by using a case study in the computational material design application. The aim was to find an efficient structure of DNN and ANFIS models to predict the fracture energy of polymer/nanoparticles composites (PNCs). The conventional analytical predictor models of PNCs show a complex and nonlinear toughening mechanism dominated by different parameters. Thus, the addressed ML models were trained to establish the inherent relation between five inputs and the fracture energy based on dataset gathered from the literature. The parameters include the percent of the volume fraction of the nanoparticles, their size, the fracture energy of the bulk epoxy polymer, its Young's modulus, and its yield strength. The dataset was divided into two groups: the training dataset used to build the model and the testing one used to validate the performance and to prevent overfitting during learning iterations. The mean squared error (MSE), the coefficient of determination (R^2), and the probability distribution of the relative error were employed to compare and to measure the performance of the optimized DNN and ANFIS models. The results indicated that the optimized DNN not only surpassed the classical one layer ANN model as expected but also yielded better prediction accuracy compared to ANFIS. The DNN with three and six hidden layers has shown the highest performance. However, it was found that as the number of the layers increases, the network structure becomes larger and the complexity of the algorithm increases. Although the results are limited to the application case study, the method can be applied to variant applications. In the future, we will extend the presented work to optimize non-supervised ML models in finding the solution of the partial differential equations.

Acknowledgements Open Access funding provided by Projekt DEAL. This study was funded by Alexander von Humboldt-Stiftung (Grant number Sofja Kovalevskaja 2015).

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Kw Chau (2017) Use of meta-heuristic techniques in rainfall-runoff modelling. Water 9(3):186. https://doi.org/10.3390/w9030186
2. Moazenzadeh R, Mohammadi B, Shamshirband S, Kw Chau (2018) Coupling a firefly algorithm with support vector regression to predict evaporation in northern iran. Eng Appl Comput Fluid Mech 12(1):584–597. https://doi.org/10.1080/19942060.2018.1482476
3. Najafi B, Faizollahzadeh Ardabili S, Shamshirband S, Kw Chau, Rabczuk T (2018) Application of ANNs, ANFIS and RSM to estimating and optimizing the parameters that affect the yield and cost of biodiesel production. Eng Appl Comput Fluid Mech 12(1):611–624. https://doi.org/10.1080/19942060.2018.1502688
4. Kononenko I, Kukar M (2007) Machine learning and data mining: introduction to principles and algorithms. Horwood Publishing, Cambridge
5. Michalski RS, Carbonell JG, Mitchell TM (2013) Machine learning: an artificial intelligence approach. Springer, Berlin
6. LeCun Y, Bengio Y, Hinton G (2015) Deep learning. Nature 521(7553):436. https://doi.org/10.1038/nature14539
7. Juang CF (2004) A hybrid of genetic algorithm and particle swarm optimization for recurrent network design. IEEE Trans Syst Man Cybern Part B (Cybern) 34(2):997–1006
8. Momeni E, Nazir R, Armaghani DJ, Maizir H (2014) Prediction of pile bearing capacity using a hybrid genetic algorithm-based ANN. Measurement 57:122–131. https://doi.org/10.1016/j.measurement.2014.08.007
9. Manngård M, Kronqvist J, Böling JM (2018) Structural learning in artificial neural networks using sparse optimization. Neurocomputing 272:660–667. https://doi.org/10.1016/j.neucom.2017.07.028
10. Sarkheyli A, Zain AM, Sharif S (2015) Robust optimization of ANFIS based on a new modified GA. Neurocomputing 166:357–366. https://doi.org/10.1016/j.neucom.2015.03.060
11. Sharafati A, Haghbin M, Motta D, Yaseen ZM (2019) The application of soft computing models and empirical formulations for hydraulic structure scouring depth simulation: a comprehensive review, assessment and possible future research direction. Arch Comput Methods Eng. https://doi.org/10.1007/s11831-019-09382-4
12. Fazilat H, Ghatarband M, Mazinani S, Asadi Z, Shiri M, Kalaee M (2012) Predicting the mechanical properties of glass fiber reinforced polymers via artificial neural network and adaptive neuro-fuzzy inference system. Comput Mater Sci 58:31–37. https://doi.org/10.1016/j.commatsci.2012.01.012
13. Shabanzadeh P, Senu N, Shameli K, Tabar MM (2013) Artificial intelligence in numerical modeling of silver nanoparticles prepared in montmorillonite interlayer space. J Chem. https://doi.org/10.1155/2013/305713
14. Mesbahi AH, Semnani D, Khorasani SN (2012) Performance prediction of a specific wear rate in epoxy nanocomposites with various composition content of polytetrafluoroethylen (PTFE), graphite, short carbon fibers (CF) and nano-tio2 using adaptive neuro-fuzzy inference system (ANFIS). Compos Part B Eng 43(2):549–558. https://doi.org/10.1016/j.compositesb.2011.11.026
15. Hamdia KM, Lahmer T, Nguyen-Thoi T, Rabczuk T (2015) Predicting the fracture toughness of PNCs: a stochastic approach based on ANN and ANFIS. Comput Mater Sci 102:304–313. https://doi.org/10.1016/j.commatsci.2015.02.045
16. Fotovatikhah F, Herrera M, Shamshirband S, Kw Chau, Faizollahzadeh Ardabili S, Piran MJ (2018) Survey of computational intelligence as basis to big flood management: challenges, research directions and future work. Eng Appl Comput Fluid Mech 12(1):411–437. https://doi.org/10.1080/19942060.2018.1448896
17. Faizollahzadeh Ardabili S, Najafi B, Shamshirband S, Minaei Bidgoli B, Deo RC, Kw Chau (2018) Computational intelligence approach for modeling hydrogen production: a review. Eng Appl Comput Fluid Mech 12(1):438–458. https://doi.org/10.1080/19942060.2018.1452296
18. Yaseen ZM, Sulaiman SO, Deo RC, Chau KW (2019) An enhanced extreme learning machine model for river flow forecasting: State-of-the-art, practical applications in water resource engineering area and future research direction. J Hydrol 569:387–408. https://doi.org/10.1016/j.jhydrol.2018.11.069
19. Haykin S (1994) Neural networks: a comprehensive foundation. Prentice Hall PTR, Upper Saddle River
20. Schmidhuber J (2015) Deep learning in neural networks: an overview. Neural Netw 61:85–117. https://doi.org/10.1016/j.neunet.2014.09.003
21. Hamdia KM, Ghasemi H, Bazi Y, AlHichri H, Alajlan N, Rabczuk T (2019) A novel deep learning based method for the computational material design of flexoelectric nanostructures with topology optimization. Finite Elem Anal Des 165:21–30. https://doi.org/10.1016/j.finel.2019.07.001
22. Rafiq M, Bugmann G, Easterbrook D (2001) Neural network design for engineering applications. Comput Struct 79(17):1541–1552. https://doi.org/10.1016/S0045-7949(01)00039-6
23. Hagan MT, Menhaj MB (1994) Training feedforward networks with the marquardt algorithm. IEEE Trans Neural Netw 5(6):989–993. https://doi.org/10.1109/72.329697
24. Suratgar AA, Tavakoli MB, Hoseinabadi A (2005) Modified levenberg-marquardt method for neural networks training. World Acad Sci Eng Technol 6(1):46–48. https://doi.org/10.5281/zenodo.1333881
25. Ross TJ (2010) Fuzzy logic with engineering applications. Wiley, Hoboken
26. Jang JSR (1993) Anfis: adaptive-network-based fuzzy inference system. IEEE Trans Syst Man Cybern 23(3):665–685. https://doi.org/10.1109/21.256541
27. Takagi T, Sugeno M (1985) Fuzzy identification of systems and its applications to modeling and control. IEEE Trans Syst Man Cybern 1:116–132. https://doi.org/10.1109/TSMC.1985.6313399
28. Morrison DR, Jacobson SH, Sauppe JJ, Sewell EC (2016) Branch-and-bound algorithms: a survey of recent advances in searching, branching, and pruning. Discrete Optim 19:79–102. https://doi.org/10.1016/j.disopt.2016.01.005
29. Hashemi SM, Rahmani I (2018) Numerical comparison of the performance of genetic algorithm and particle swarm optimization in excavations. Civil Eng J 4(9):2186–2196. https://doi.org/10.28991/cej-03091149

30. Karavasilis M, Tsakiroglou CD (2019) Synthesis of aqueous suspensions of zero-valent iron nanoparticles (nZVI) from plant extracts: experimental study and numerical modeling. Emerg Sci J 3(6):344–360. https://doi.org/10.28991/esj-2019-01197
31. Holland JH et al (1992) Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. MIT press, Cambridge
32. Deep K, Singh KP, Kansal ML, Mohan C (2009) A real coded genetic algorithm for solving integer and mixed integer optimization problems. Appl Math Comput 212(2):505–518. https://doi.org/10.1016/j.amc.2009.02.044
33. Goldberg DE (1989) Genetic algorithms in search. Optimization and machine learning. Addison-Wesley, Boston
34. Cui C, Guo C, Wang F (2019) Fatigue analysis for void repair of cement concrete pavement with under slab by polymer grouting. Civil Eng J 5(7):1452–1464. https://doi.org/10.28991/cej-2019-03091344
35. Scharnberg ARA, de Loreto AC, Alves AK (2020) Optical and structural characterization of Bi2FexNbO7 nanoparticles for environmental applications. Emerg Sci J 4(1):11–17. https://doi.org/10.28991/esj-2020-01205
36. Argon A, Cohen R (2003) Toughenability of polymers. Polymer 44(19):6013–6032. https://doi.org/10.1016/S0032-3861(03)00546-9
37. Thostenson ET, Li C, Chou TW (2005) Nanocomposites in context. Compos Sci Technol 65(3):491–516. https://doi.org/10.1016/j.compscitech.2004.11.003
38. Hamdia KM, Zhuang X, He P, Rabczuk T (2016) Fracture toughness of polymeric particle nanocomposites: evaluation of models performance using bayesian method. Compos Sci Technol 126:122–129. https://doi.org/10.1016/j.compscitech.2016.02.012
39. Huang Y, Kinloch A (1992) Modelling of the toughening mechanisms in rubber-modified epoxy polymers. part II a quantitative description of the microstructure-fracture property relationships. J Mater Sci 27(10):2763–2769. https://doi.org/10.1007/BF00540703
40. Williams J (2010) Particle toughening of polymers by plastic void growth. Compos Sci Technol 70(6):885–891. https://doi.org/10.1016/j.compscitech.2009.12.024
41. Quaresimin M, Salviato M, Zappalorto M (2014) A multi-scale and multi-mechanism approach for the fracture toughness assessment of polymer nanocomposites. Compos Sci Technol 91:16–21. https://doi.org/10.1016/j.compscitech.2013.11.015
42. Lauke B (2017) Fracture toughness modelling of polymers filled with inhomogeneously distributed rigid spherical particles. Express Polym Lett 11(7):545. https://doi.org/10.3144/expresspolymlett.2017.52
43. Silani M, Ziaei-Rad S, Esfahanian M, Tan V (2012) On the experimental and numerical investigation of clay/epoxy nanocomposites. Compos Struct 94(11):3142–3148. https://doi.org/10.1016/j.compstruct.2012.04.033
44. Hamdia KM, Rabczuk T (2018) Key parameters for fracture toughness of particle/polymer nanocomposites; sensitivity analysis via xfem modeling approach. In: Wahab MA (ed) Proceedings of the 7th International Conference on fracture, fatigue and wear. Springer, Berlin, pp 41–51. https://doi.org/10.1007/978-981-13-0411-8_4
45. Msekh MA, Cuong N, Zi G, Areias P, Zhuang X, Rabczuk T (2018) Fracture properties prediction of clay/epoxy nanocomposites with interphase zones using a phase field model. Eng Fract Mech 188:287–299. https://doi.org/10.1016/j.engfracmech.2017.08.002
46. Zamanian M, Mortezaei M, Salehnia B, Jam J (2013) Fracture toughness of epoxy polymer modified with nanosilica particles: particle size effect. Eng Fract Mech 97:193–206. https://doi.org/10.1016/j.engfracmech.2012.10.027
47. Zappalorto M, Pontefisso A, Fabrizi A, Quaresimin M (2015) Mechanical behaviour of epoxy/silica nanocomposites: experiments and modelling. Compos Part A Appl Sci Manuf 72:58–64. https://doi.org/10.1016/j.compositesa.2015.01.027
48. Zuo K, Blackman B, Williams J, Steininger H (2015) The mechanical behaviour of ZnO nano-particle modified styrene acrylonitrile copolymers. Compos Sci Technol 113:9–18. https://doi.org/10.1016/j.compscitech.2015.02.014
49. Carolan D, Ivankovic A, Kinloch A, Sprenger S, Taylor A (2017) Toughened carbon fibre-reinforced polymer composites with nanoparticle-modified epoxy matrices. J Mater Sci 52(3):1767–1788. https://doi.org/10.1007/s10853-016-0468-5