Characterizing Fracture Stress of Defective Graphene Samples using Shallow and Deep Artificial Neural Networks

M. A. N. Dewapriya[1]*, R. K. N. D. Rajapakse[1,3], and W. P. S. Dias[2]
[1]School of Engineering Science, Simon Fraser University, Burnaby, Canada
[2]Department of Civil Engineering, University of Moratuwa, Moratuwa, Sri Lanka
[3] Sri Lanka Institute of Information Technology, Malabe, Sri Lanka

Abstract

Advanced machine learning methods could be useful to obtain novel insights into some challenging nanomechanical problems. In this work, we employed artificial neural networks to predict the fracture stress of defective graphene samples. First, shallow neural networks were used to predict the fracture stress, which depends on the temperature, vacancy concentration, strain rate, and loading direction. A part of the data required to model the shallow networks was obtained by developing an analytical solution based on the Bailey durability criterion and the Arrhenius equation. Molecular dynamics (MD) simulations were also used to obtain some data. Sensitivity analysis was performed to explore the features learnt by the neural network, and their behaviour under extrapolation was also investigated. Subsequently, deep convolutional neural networks (CNNs) were developed to predict the fracture stress of graphene samples containing random distributions of vacancy defects. Data required to model CNNs was obtained from MD simulations. Our results reveal that the neural networks have a strong ability to predict the fracture stress of defective graphene under various processing conditions. In addition, this work highlights some advantages as well as limitations and challenges in using neural networks to solve complex problems in the domain of computational materials design.

Keywords: Deep learning; neural networks; molecular dynamics; defective graphene; fracture stress.

*Corresponding author. Tel: 778 707-2678 E-mail: mandewapriya@sfu.ca (Nuwan Dewapriya)

1. Introduction

Computational design and characterization of nanomaterials are often limited by the available computational power. For example, first principle methods such as density functional theory have limited applicability to fracture characterization at the nanoscale due to high computational cost [1]. On the other hand, even computationally efficient continuum-based numerical methods such as finite element methods become prohibitively expensive when they are used to study complicated fracture mechanics problems such as crack propagation in the presence of interacting microcracks [2,3]. Moreover, such continuum-based methods are not directly applicable to the corresponding problems at the nanoscale because the surface energy effects and discrete nature of the underlying atomic structure become important as the characteristic dimension reaches down to a few nanometres [4–6]. On the other hand, it appears that molecular dynamics (MD) simulations strike a balance between computational cost and accuracy [7]. Even though MD simulations can be used to generate a large amount of data, the existing continuum-based theoretical frameworks are unable to properly characterize this data due to the fact that continuum methods often preclude the influence of discreteness and surface effects. For example, the underlying crystal structure of graphene has a significant influence on the stress concentration at an existing defect/crack [8,9] and its propagation [10,11], which is difficult to capture within the framework of classical continuum mechanics.

There has been a recent interest in data-driven computing where empirical material modelling is eliminated by directly using the experimental data for computations [12,13]. On the other hand, machine learning techniques, which can be used to extract underlying features of a large data set [14], could provide an opportunity to gain novel insights into computational materials design [15,16]. Machine learning methods such as neural networks have been successfully used to predict crack growth in brittle materials [2,3,17]. In recent years, deep learning (e.g., the use of deep neural networks) has attracted significant attention from engineers and scientists due to its extraordinary success in computer vision problems [18]. However, the application of deep learning to the problems in computational materials science is still in its infancy [19]. Deep neural networks have been employed to develop a new method for numerical integration of finite element stiffness matrices [20] and to solve nonlinear dynamic problems [21,22]. Moreover, deep learning has been used to predict structure-property relationships of ceramics and composites [23,24], and it has also

been demonstrated that generative deep learning models can be successfully employed to generate artificial material samples, which can be used to develop predictive models concerning structure-property relationships [25]. In addition, deep neural networks have been employed to solve challenging nanoscale problems such as predicting mechanical properties of graphene under various processing conditions [26] and interfacial thermal resistance between graphene and hexagonal boron nitride [27].

Several recent studies have demonstrated that the computation of interatomic potential energy could be replaced with properly trained neural networks at a significantly lower computational power [28–30]. These preliminary studies suggest that the application of deep learning techniques could significantly revolutionize atomistic modelling of the mechanical behaviour of materials. For example, deep convolutional networks, which have been very successful in image recognition, could be used to obtain novel insights into complex nanoscale problems such as fracture and crack propagation in the presence of interacting nanocracks, which are computationally expensive to solve using atomistic simulations. Such studies are important to the emerging topic of tailoring of the physical properties through the topological design of nanomaterials with defects. In this regard, it is noted that deep learning has not yet been employed to investigate the effect of defect distribution on the fracture stress of nanomaterials. In this work, we employ both shallow and deep neural networks to predict the fracture stress of defective graphene samples under various processing conditions, including a detailed investigation into the effect of defect distribution on the fracture stress of graphene. Section 2 provides a brief overview of neural networks, molecular dynamics, and analytical modelling of the fracture stress of defective graphene. In Section 3, modelling of shallow and deep neural networks to predict fracture stress of defective graphene is presented.

2. Atomistic Modelling and Deep Learning

2.1 *Artificial Neural Networks and Deep Learning*

Artificial neural networks contain a set of systematically interconnected simple units called neurons, which are inspired by the neurons in the biological nervous systems (see Fig. 1). Strengths of the connections between individual neurons, called weights (w_i), determine the ability of the

neural network to represent a function of interest. The process of adjusting the weights is carried out by comparing the network's outputs and the target outputs, until the network output matches the target with a reasonable accuracy. The process of adjusting the weights is called training or learning of a network. The output of the individual neuron shown in Fig. 1 can be expressed as $y = f(b + \sum_{i=1}^{n} w_i\, x_i)$, where b is a bias, which offsets the nonlinear activation function f [31]. In general, the number of training examples should be comparable to the total number of learnable parameters of the network (e.g., weights, biases). However, in the case of deep neural networks, it is often not practical to keep the number of training examples and the total number of learnable parameters comparable due to the large number of learnable parameters associated with deep neural networks, which can be in the order of 100 millions [32–34].

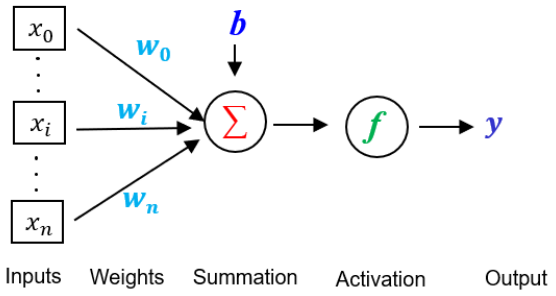

Figure 1 Schematic representation of an artificial neuron.

It has been proved that a neural network with one hidden layer is capable of simulating a nonlinear function up to a desired degree of accuracy [35,36]. Figure 2 shows a typical neural network, containing one hidden layer, which we used to predict the fracture stress of defective graphene samples. The four inputs to the network (i.e. x_1 to x_4) are the temperature, vacancy concentration, directional constant, and strain rate. The hidden layer contains n neurons, where the best value for n was found by trial and error. The output (O_1) is the fracture stress of the sample. In order to facilitate the learning of the network, the input and output data are transformed by linear mapping of the respective minimum and maximum values to be -1 and 1, respectively.

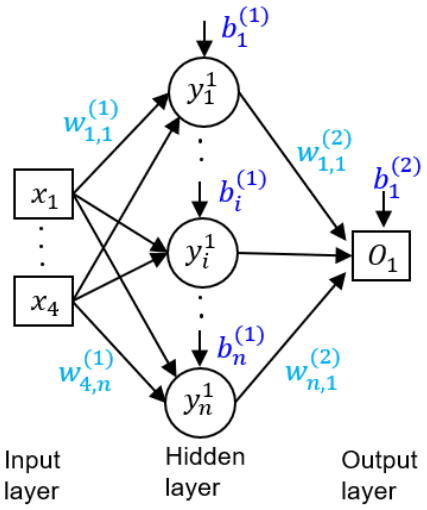

Figure 2 A typical neural network used to predict the fracture stress of defective graphene.

The output of the network shown in Fig. 2 can be expressed as

$$O_1 = f_a^o \left[b_1^{(2)} + \sum_{j=1}^{n} w_{j,1}^{(2)} f_a^{(h)} \left(b_j^{(1)} + \sum_{i=1}^{4} w_{i,j}^{(1)} x_i \right) \right] \quad (1)$$

Where, $w_{i,j}^{(k)}$ is the weight associated with the connection between node i in the layer k and the node j in the layer k+1. For example, $w_{3,4}^{(1)}$ is the weight of the connection between node 3 in layer 1 and node 4 in layer 2. The functions f_a^h and f_a^o are the activation functions. We used the sigmoid function for the hidden layer (i.e. f_a^h) and a linear function for the output layer (i.e. f_a^o). The biases $b_j^{(k)}$ are used to offset the activation functions.

The initial weights and biases of the network were randomly assigned, and their values were then iteratively adjusted using the error backpropagation method in such a way that the total error is minimized [37]. We used the Levenberg-Marquardt method to compute optimized weights and biases [38]. Once the performance of the network on training data is satisfactory, the network is evaluated using a data set that has not been used to train the network, which is called test data. The optimum number of neurons in the hidden layer is selected based on the performance of the network on the test data. All neural networks were modelled in MATLAB.

2.1.1 *Deep Convolutional Neural Networks*

Convolutional neural networks (CNNs) are a special type of neural networks that use convolution operations in at least one of its layers. These networks are used to process data arranged into a fixed grid-type topology [18,33,39]. A brief overview of CNNs is provided in this section; a comprehensive review on the topic can be found in [33]. CNNs are widely used for classification problems in computer vision; however, they could be successfully used for regression problems as well.

Figure 3 shows the architecture of a typical CNN with a single convolutional layer. Input to a CNN is generally an image, which is an array of number (e.g., a 2D array for a grey-scale image). In this study, a typical input to a CNN is an image containing the distribution of vacancy defects in a graphene sample and the required output is the fracture stress of the sample. The convolutional layer computes output from a set of neurons (called a filter) that is connected to local regions of the input image. Each filter convolves through the entire input space and produces an activation map. The number of activation maps (i.e. output of a convolutional layer) depends on the number of filters assigned to the layer.

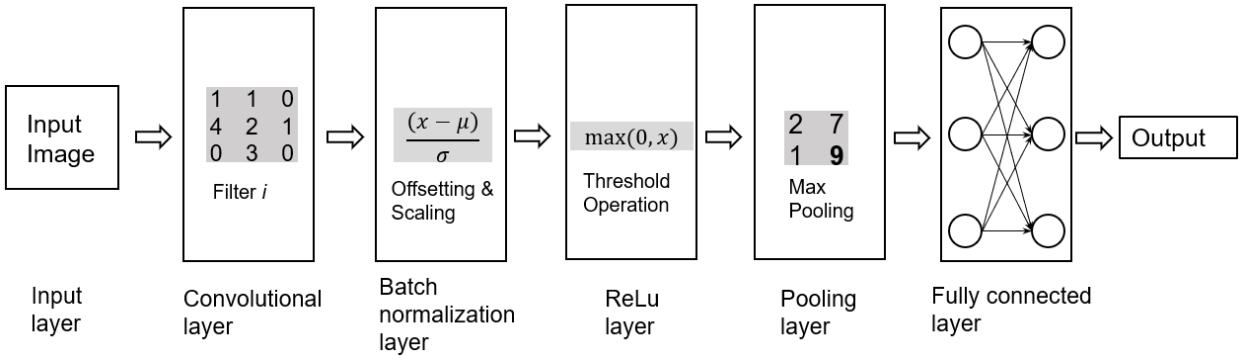

Figure 3 The architecture of a typical convolutional neural network containing a single convolutional layer.

The activation maps from the convolutional layer are generally normalized by a batch normalization layer in order to facilitate the learning process. The normalization of the input channel is carried out across each mini-batch, which is a subset of the training data. Output of the normalization layer can be expresses as $\gamma[(x - \mu)/\sigma + \beta]$, where x is the input to the layer, μ and σ

are the mean and standard deviation of the mini-batch, respectively; β and γ are learnable parameters, which offset and scale the output, respectively. The normalized output then goes through a nonlinear activation layer. We employed the rectified linear unit (ReLU) activation function, which is one of the most widely used functions for CNNs [33]. The ReLU function performs a threshold operation on each channel, and its functional form can be given as $max(x,0)$, where x is the input to the activation function.

The pooling layer is used to reduce the spatial size of the activation maps, resulting in a reduced number of learnable parameters in the network. The fully connected layer is similar to a layer in a classical shallow neural network, where all neurons in a layer are connected to all the neurons in the previous layer. The fully connected layer combines all the local features of the input image, identified by the preceding layers in the form of activation maps, to identify unique features at higher level. Training and testing of a deep CNN is similar to the case of shallow neural networks, which explained above.

2.1.2 *Transfer Learning*

As demonstrated in Fig. 4, the middle layers of a CNN extract some special features in the input data; and the final fully connected layers convert those features into a desired output (e.g., classification or regression). Therefore, the convolutional layers remain the same for both classification and regression networks, which allows us to use a pretrained CNN on a different data set to solve other problems; this is known as transfer learning [40]. Transfer learning is widely considered to be the first step of solving a problem using a deep CNN because it readily provides a good estimation of the performance of a typical CNN; and more importantly, transfer learning requires a smaller number of training examples. In addition, transfer learning provides guidance in developing a new CNN architecture for a given problem.

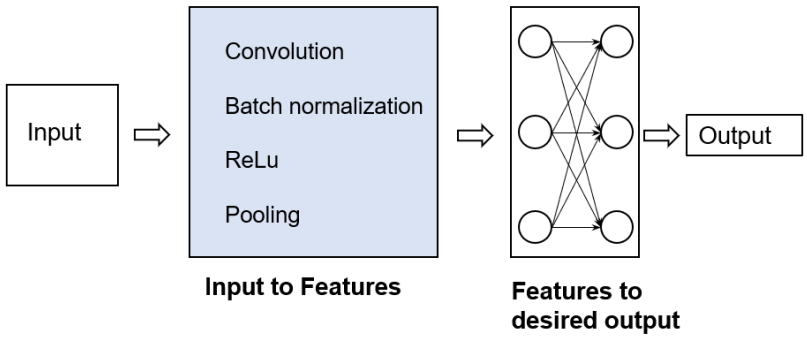

Figure 4 A high level representation of the function of a typical CNN.

Transfer learning is generally performed using a well-recognised pretrained deep CNNs such as AlexNet and ResNet, which have been trained using millions of images to identify 1000 classes of objects. In 2012, AlexNet significantly outperformed the other algorithms of object recognition, and generated great interest within the computer vision community for adopting deep learning for object recognition [32]. AlexNet contains 8 layers and the number of associated learnable parameters is approximately 62 million. The recently developed residual network (ResNet) uses new residual learning algorithm to train much deeper CNNs [41]; for example, ResNet152 has 152 layers. and the number of learnable parameters is approximately similar to that of AlexNet. In this work, we conducted transfer learning using AlexNet to predict the fracture stress of defective graphene samples; subsequently, we developed a new CNN as explained in the following section.

2.1.3 *Modelling of a New CNN*

The developed new CNN contains 30 layers (see Fig. 5) and the distribution of learnable parameters are given in Table 1. The network has eight convolutional layers (C) and each layer is followed by a batch normalization (BN) layer and a ReLU layer (R). Four average pooling layers (AP) were used to reduce the activation volume and a dropout layer (D) was used to prevent overfitting. The total number of learnable parameters of the network is 1998.

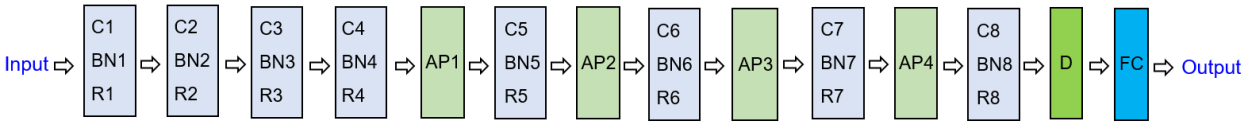

Figure 5 Architecture of the developed CNN. Details of each layer is given in Table 1.

Table 1 Parameters of individual layers of the developed CNN.

Name	Type	Activations	Learnable	Total Learnable
Input	Input	224×224×1	-	0
C1	Convolution	224×224×3	Weights: 3×3×1×3 Bias: 1×1×3	30
BN1	Batch Normalization	224×224×3	Offset: 1×1×3 Scale: 1×1×3	6
R1	ReLu	224×224×3	-	0
C2 to C4	Convolution	224×224×3	Weights: 3×3×3×3 Bias: 1×1×3	84
BN2 to RN4	Batch Normalization	224×224×3	Offset: 1×1×3 Scale: 1×1×3	6
R2 to R4	ReLu	224×224×3	-	0
AP1	Average Pooling	112×112×3	-	0
C5	Convolution	112×112×4	Weights: 3×3×3×4 Bias: 1×1×4	112
BN5	Batch Normalization	112×112×4	Offset: 1×1×4 Scale: 1×1×4	8
R5	ReLu	112×112×4	-	0
AP2	Average Pooling	56×56×4	-	0
C6	Convolution	56×56×4	Weights: 3×3×4×4 Bias: 1×1×4	148
BN6	Batch Normalization	56×56×4	Offset: 1×1×4 Scale: 1×1×4	8
R6	ReLu	56×56×4	-	0
AP3	Average Pooling	28×28×4	-	0

C7	Convolution	28×28×5	Weights: 3×3×4×5 Bias: 1×1×5	185
BN7	Batch Normalization	28×28×5	Offset: 1×1×5 Scale: 1×1×5	10
R7	ReLu	28×28×5	-	0
AP4	Average Pooling	14×14×5	-	0
C8	Convolution	14×14×5	Weights: 3×3×5×5 Bias: 1×1×5	230
BN8	Batch Normalization	14×14×5	Offset: 1×1×5 Scale: 1×1×5	10
R8	ReLu	14×14×5	-	0
D	Dropout	14×14×5	-	0
FC	Fully Connected	1×1×1	Weights: 1×980 Bias: 1×1	981
Output	Output	-	-	0

2.2 *Analytical Modelling of Fracture Strength*

Part of the data required to train the shallow neural networks was obtained by developing an analytical solution based on the Bailey durability criterion and the Arrhenius equation. We earlier developed a numerical model to compute the fracture stress of graphene containing a random distribution of single vacancies [42], which was later extended to develop an analytical model to study the influence of hydrogen functionalization on the fracture stress [43]. In the current work, we combine our previous work to arrive at an analytical model to evaluate the fracture stress of graphene samples containing a random distribution of single vacancies. The model combines the Arrhenius equation [44] and the Bailey durability criterion [45] to achieve a generalized analytical solution where the predicted fracture stress is a function of temperature, strain rate, vacancy concentration, and loading direction. The development of the analytical solution is outlined in this section.

The Bailey durability criterion (see Eq. (2)) provides a theoretical framework to compute the temperature dependent lifetime of a material sample [45,46].

$$\int_0^{t_f} \frac{dt}{\tau(T,t)} = 1 \tag{2}$$

where, t_f is the lifetime; $\tau(T,t)$ is the durability function that depends on time (t) and temperature (T), which is generally an empirical function [46]. In the absence of an empirical durability function for graphene, a durability function in the form of Arrhenius equation can be defined [42,43]. The Arrhenius equation [44] expresses temperature dependent rate of a chemical reaction (k) as

$$k = A \times \exp\left(\frac{\Delta E}{K_B T}\right) \tag{3}$$

where A is a constant that depends on the material; ΔE is the activation energy barrier and k_B is the Boltzmann constant. This general form of the Arrhenius equation can be used to define a durability function for graphene as

$$\tau_g(T,t) = \frac{\tau_0}{n} \times \exp\left(\frac{U_0/\beta - v\gamma\sigma(t)}{k_B T}\right) \tag{4}$$

where τ_0 is the vibration period of the atoms; n is the number of bonds in the sample; U_0 is the bond dissociation energy, which is approximately 4.95 eV for a carbon-carbon bond in graphene; v is the representative volume of a carbon atom in graphene, which is approximately 8.6 Å^3; and γ is the directional constant that represents directional dependence of the fracture stress. Following the work of Young et al. [47], the constant γ for pristine graphene can approximated as $\cos(\theta)$, where the angle θ is measured from the armchair direction towards to the chiral direction of interest. For example, θ is 30˚ for the zigzag direction. However, the presence of multiple vacancies remarkably alters the chirality dependent fracture stress. Therefore, based on our MD simulations conducted at a temperature of 300 K, we selected γ to be $(1.08 + \cos\theta)/2$.

The stress at time t in a graphene sample can be expressed in terms of the strain rate, $\dot{\varepsilon}$, as

$$\sigma(t) = a(\dot{\varepsilon}t) + b(\dot{\varepsilon}t)^2 \tag{5}$$

where *a* and *b* are the second and the third order elastic moduli, respectively. The values of *a* and *b* were computed by performing regression analysis of the stress-strain curves obtained from MD simulations, and the corresponding values of *a* and *b* are 1.11 TPa and -3.20 TPa for armchair graphene.

The constant β represents the reduction of the activation energy barrier due to the presence of vacancies, which is related to the vacancy concentration (α) as

$$\beta = \begin{cases} 1, & \alpha = 0 \\ 0.165\alpha + 1.13, & \alpha > 0 \end{cases} \tag{6}$$

It should be noted that β is discontinuous when α equals zero, which is due to the fact that even a single vacancy (i.e. α is very small) has a remarkable influence on the activation energy barrier, resulting a large reduction in the fracture stress [42].

The governing equation of Bailey's principle for a graphene sheet can now be expressed as

$$\int_0^{t_f} exp\left(\lambda\sigma(t) - \frac{U_0}{\beta\upsilon\gamma}\right) = \frac{\tau_0}{n} \tag{7}$$

where $\lambda = \upsilon\gamma/k_B T$. By substituting for $\sigma(t)$, Eq. (7) can be expressed as

$$\int_0^{t_f} exp\left(\lambda\left[b\left(\dot{\varepsilon}t + \frac{a}{2b}\right)^2 - \left(\frac{U_0}{\beta\upsilon\gamma} + \frac{a^2}{4b}\right)\right]\right) = \frac{\tau_0}{n} \tag{8}$$

Then, using the definition of the error function [48], where $erf(x) = 2/\sqrt{\pi}\int_0^x e^{-t^2}dt$, Eq. (8) can be solved for the life time t_f as

$$t_f = \left(erf^{-1}\left\{\sqrt{\frac{-b}{\pi}}\left(\frac{2\lambda\tau_0\dot{\varepsilon}}{n}\right) exp\left[\lambda^2\left(\frac{U_0}{\beta\upsilon\gamma} + \frac{a^2}{4b}\right)\right] - erf(\chi)\right\} + \chi\right)\Big/\sqrt{-b}\lambda\dot{\varepsilon} \tag{9}$$

where erf^{-1} is the inverse of the error function and $\chi = \lambda a/\sqrt{-4b}$. When t_f is known, the fracture stress, $\sigma(t_f)$, can be obtained from Eq. (5).

Our previous atomistic model to compute the fracture stress of graphene containing single

vacancies [42] has to be solved numerically, which may be viewed as finding a numerical solution to Eq. (7). In addition, the previous model has only been parameterized to compute fracture stress along the armchair and zigzag directions. The current analytical model is more convenient to use, and it predicts the fracture stress of the graphene sample when loaded along any chiral direction.

2.3 *Molecular Dynamics Simulations*

Data required to train deep convolutional networks was obtained from MD simulations. Moreover, the performance of the shallow networks as well as the analytical model were evaluated against the corresponding MD simulation results. The uniaxial tensile tests of the samples were simulated using the LAMMPS MD simulator [49] and the AIREBO potential [50], which contains three sub-potentials: the Lennard-Jones (LJ), torsional, and reactive empirical bond order (REBO) potentials. The LJ and torsional potentials evaluate the energy due to the van der Waals and torsional interactions between atoms, respectively. The REBO potential expresses energy stored in a bond between atom i and atom j as

$$E_{ij}^{REBO} = f(r_{ij})\left[V_{ij}^{R} - b_{ij}V_{ij}^{A}\right] \tag{10}$$

where V_{ij}^{R} and V_{ij}^{A} are the repulsive and attractive potentials respectively; b_{ij} is the bond order term, which modifies the attractive potential depending on the local bonding environment; r_{ij} is the distance between the atoms i and j; and $f(r_{ij})$ is the cut-off function, which limits the interatomic interactions to the nearest neighbors. In order to avoid the previously observed non-physical strain hardening, the cut-off distance of the REBO potential was modified to be 2 Å [51].

A typical graphene sample used for the MD simulations is shown in Fig. 6. The planar dimensions of the samples were 15 nm × 15 nm. Even though smaller samples (e.g. 5 nm × 5nm) are sufficient to compute the mechanical properties of pristine samples [52], relatively larger samples are required in the case of defective graphene due to remarkable size effects at the atomic scale. For example, when simulating graphene samples with cracks (i.e. a row of vacancies), length and width of the simulated graphene sheets must be kept at more than ten times the half initial crack length in order to avoid the effects of finite dimensions on the MD simulation results [10,11,53]. Even though dimensions of 10 nm × 10 nm are sufficient to simulate graphene samples containing a

vacancy concentration of 10% [54,55], we selected sample dimensions of 15 nm × 15 nm, which allows us to generate various spatial distributions of defects within the sample. The maximum vacancy concentration used in the current MD simulations is 2%, where the defects were randomly distributed.

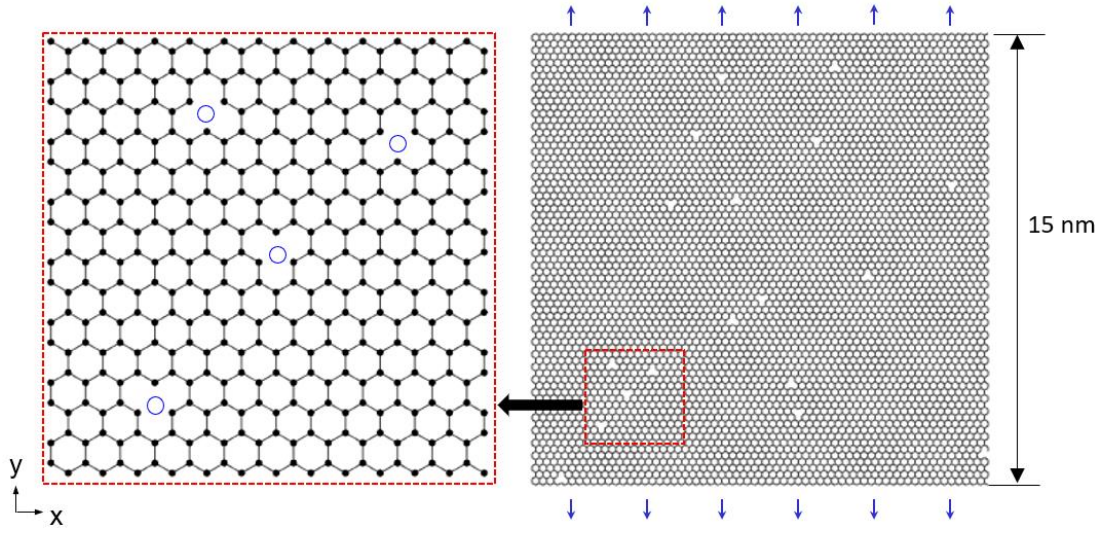

Figure 6 Molecular dynamics model of a graphene sample containing 0.2% of random vacancy defects. The blue arrows indicate the loading direction. The blue circles on the inset indicate missing atoms (i.e. vacancy defects).

Before conducting the numerical tensile tests, the simulation samples were equilibrated over a period of 25 ps, the selected time step being 0.5 fs. Even though special energy minimization techniques and subsequent relaxation over a large period is necessary for bulk materials such as polymer [56,57], we found that graphene, which is a single layer of carbon atoms, reaches equilibrium in around 10 ps. Convergence of total energy was used to identify the equilibrium state of a graphene sample. Initial displacement perturbations (~0.001 nm) were imposed on the atoms along the x-, y-, and z-directions in order to facilitate reaching their equilibrium configuration [52]. Periodic boundary conditions were implemented along the x- and y-directions. The simulations were performed with the isothermal–isobaric (NPT) ensemble, and the Nośe-Hoover thermostat was used to keep the temperature constant during the simulation. After the equilibrium period, the graphene samples were subjected to strain along the y-direction (ε_{yy}), at a rate of 0.001 ps^{-1}, until fracture. The virial theorem was used for the calculation of atomic stress [58]. The averaged virial stress tensor, σ_{ij}, is defined as follows:

$$\sigma_{ij} = \frac{1}{V}\sum_{\alpha}\left[\frac{1}{2}\sum_{\beta=1}^{N}\left(R_i^{\beta} - R_i^{\alpha}\right)F_j^{\alpha\beta} - m^{\alpha}v_i^{\alpha}v_j^{\alpha}\right] \quad (11)$$

where i and j are the directional indices (i.e. x, y, and z); α is a number assigned to an atom and β is a number assigned to neighboring atoms of α; R_i^{β} is the position of atom β along the direction i; $F_j^{\alpha\beta}$ is the force on atom α due to atom β along the direction j; m^{α} and v^{α} are the mass and the velocity of atom α respectively; and V is the total volume. In volume calculations, the thickness of graphene was assumed to be 3.4 Å [59].

Fracture of the sample was identified by a sudden drop in its stress, with the maximum value of stress defined as the fracture stress. In one of our previous publications [42], we have compared the stress-strain curves of armchair and zigzag graphene samples, both pristine and containing vacancy defects, under various temperatures. Moreover, we have previously validated our MD simulation results of graphene against available experimental results and density functional theory calculations in the context of elastic properties and fracture stress of pristine structures [9], fracture toughness [10], and surface energy [8,11].

3. Results and Discussion

3.1 *Modelling Fracture Stress*

3.1.1 *Data for Neural Network Modelling*

A sufficiently large data set is required to properly train a neural network. We obtained a part of data to train shallow networks from the analytical model developed in Section 2.2, where the inputs to the network are the temperature, vacancy concentration, loading direction, and strain rate. The span of each input variables is given in Table 2. The temperature range in Table 2 was selected to cover the range that a graphene sample could experience during synthesis of graphene based products [60,61]. The range of strain rates was selected considering the rates used in typical MD simulations and experimental work [62,63]. The only output of the network is the fracture stress of a graphene sample.

Figure 7 shows the distribution of fracture stress with respect to two selected input parameters (out of four) as predicted by the analytical model. It can be noticed that vacancy concentration has the

highest influence on the fracture stress and the influence is more significant at elevated temperatures (see Fig. 7a) and at small strain rates (see Fig. 7c). Interestingly, the loading direction does not have a significant influence on the fracture stress in the presence of vacancy defects (see Fig. 7b). Due to the limited availability of experimental results related to fracture of graphene, we validated the analytical model using MD simulation results, in the context of having previously validated our MD simulations with respect to the available experiments and first principle computations related to fracture of graphene [8–11]. In addition to the data obtained from the analytical model, we used MD simulations to obtain limited data samples to train and validate neural networks as explained in the proceeding sections.

Table 2 Ranges of input data used to train the neural network.

Input	Range
Temperature	350 K to 950 K
Vacancy concentration	0 to 4.5%
Directional constant	0.974 to 1.04
$\log_{10}$(Strain rate)	0.5 to 9.5

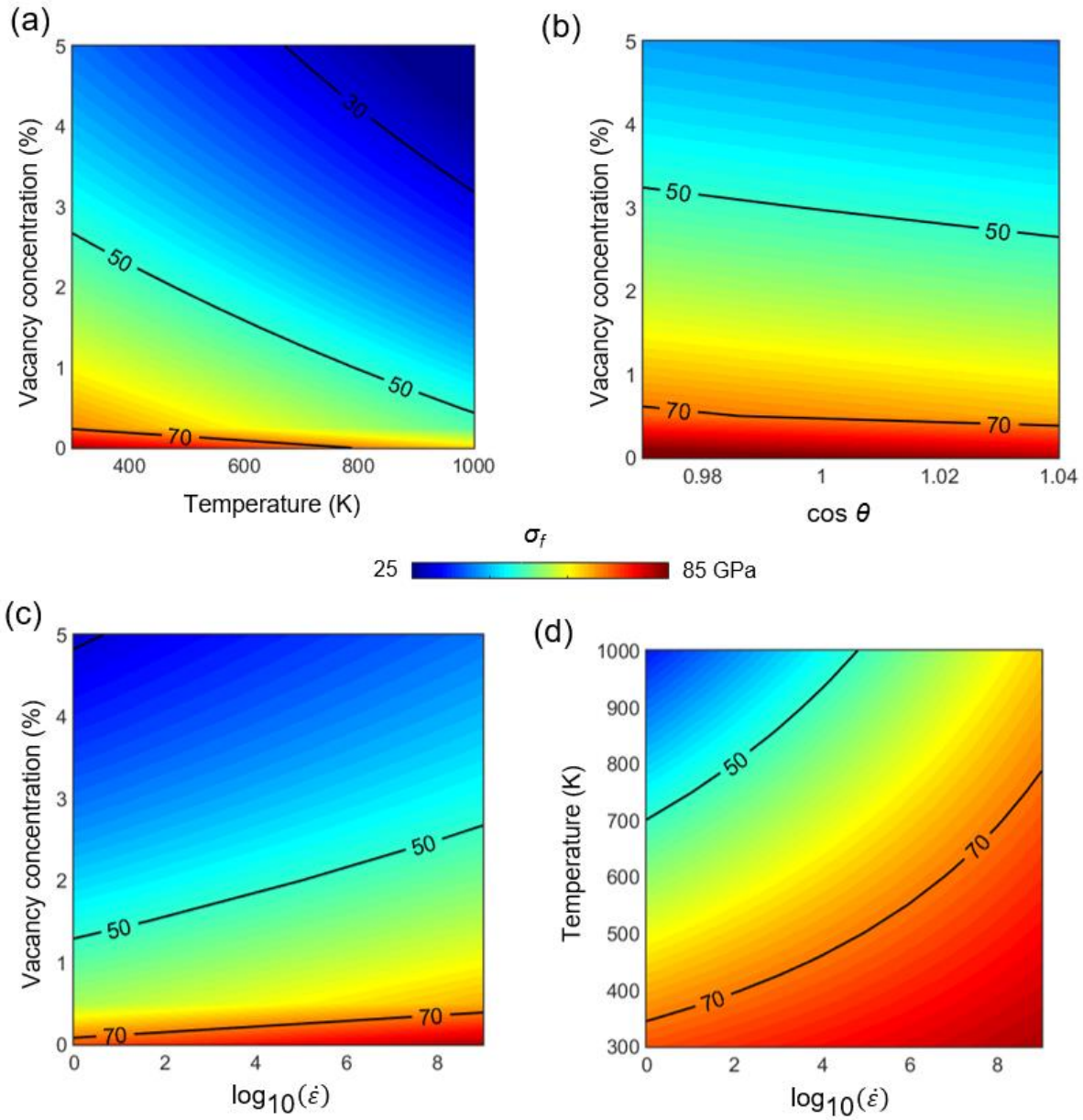

Figure 7 Variation of the fracture stress σ_f of graphene samples with two independent variables: (**a**) vacancy concentration and temperature, (**b**) vacancy concentration and chirality, where the cosine value of the chiral angle θ has been plotted on the x-axis, (**c**) vacancy concentration and strain rate, and (**d**) temperature and strain rate.

When modelling a neural network, we divided the data set into three different subsets namely: (a) training set, (b) validation set, and (c) test set. The training set was used to adjust the weights and biases of the network during the training phase and the validation set was used to measure the generalization of the network during the training, which ensures that the network does not overfit

the training data. The change in the mean square error (MSE) of the validation test during the training was used as a criterion to terminate training- i.e. the training was terminated when the MSE did not reach a new minimum in six consecutive iterations. The test set provides an independent measure of the performance of the network. From the original data set, 70% of data samples were randomly selected for the training and the remaining 30% used for validation and testing (15% each). Unless otherwise stated, similar segmentations of data are used to train, validate, and test neural networks in the ensuing sections.

3.1.2 *Modelling of Neural Networks*

First, we identified the optimum number of hidden neurons by evaluating the performance of a network by varying the number of hidden neurons; here the MSE of the test set was used to measure the performance of the network. Three data sets containing 250, 500, and 1000 samples were used for each neural network to evaluate influence of the number of training data samples on the resulting performance. The initial weights and biases are randomly assigned, and therefore each network was trained 25 times with different initial weights and biases; the network with the smallest MSE is considered to be the best network. It can be seen in Fig. 8 that a network with four hidden layers can accurately evaluate the fracture stress irrespective of the number of training samples.

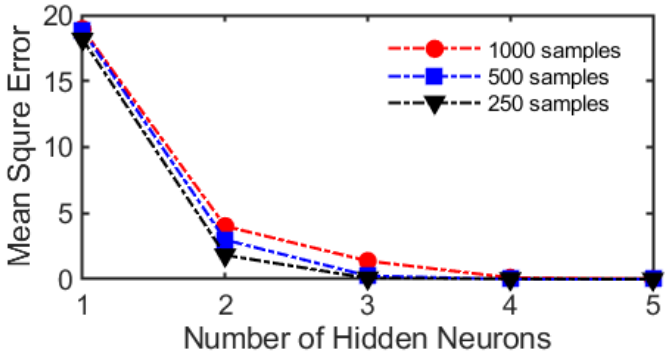

Figure 8 Variation of mean square error of the test set with the number of hidden neurons for three different number of data samples.

Based on this initial result, we selected the network with five hidden neurons containing 31 learnable parameters for further investigation. This network was modelled with 1000 data samples,

and Fig. 9 shows its performance. It can be noticed that output is almost identical to the target within the entire range of the targets, where the best fit curve to the outputs coincides with the output = target line. The largest error is of the order of 0.5 (see Fig 9c) and the percentage error less than 1%.

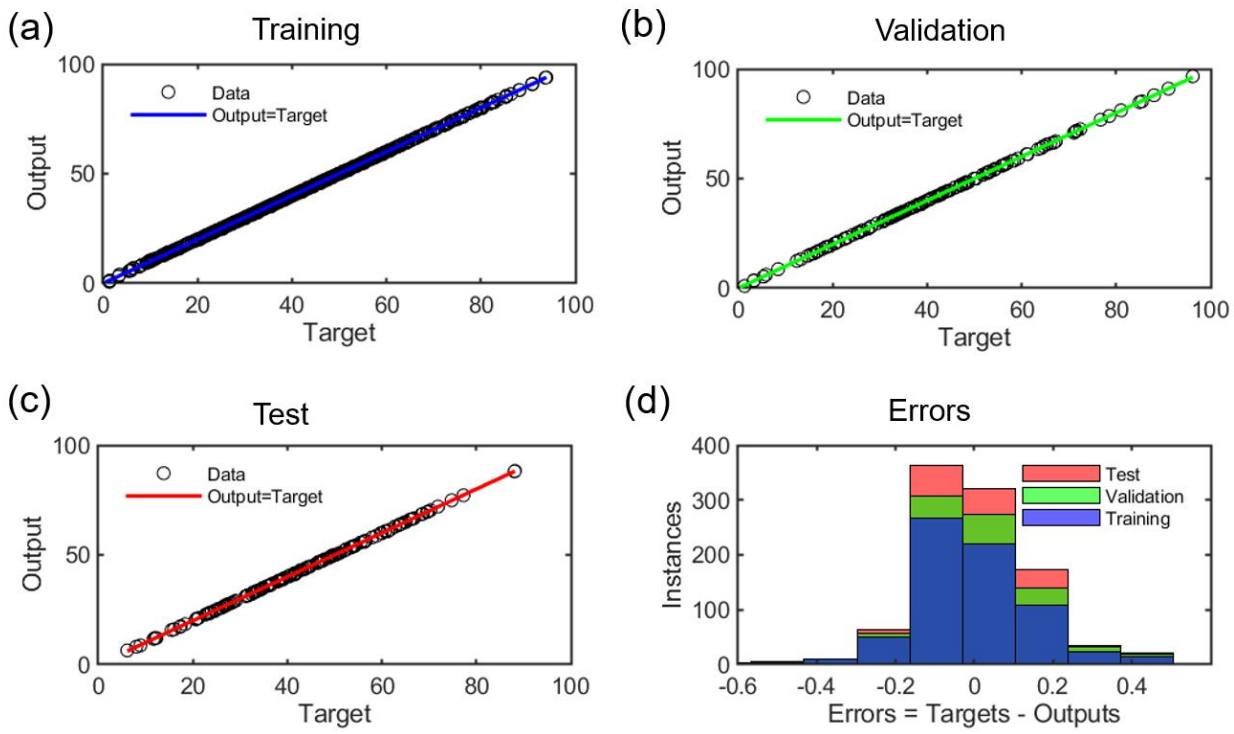

Figure 9 Performance of the network containing five hidden neurons: (**a**) to (**c**) output-target relationship of training, validation, and test sets. (**d**) The distribution of errors of the three data sets.

In order to further validate the predictions of the neural network as well as the analytical model, we compared their predictions with corresponding MD simulation results (see Fig. 10). Two vacancy concentrations were considered (i.e. 0.5% and 2%) and the loading was applied to samples along the armchair or zigzag direction. The simulation temperature was selected to be in the range from 300 K to 1500 K, and the strain rate was kept constant at 10^9 s^{-1}. Due to the fact that the fracture stress depends on the distribution of vacancies, ten MD simulations were conducted for each vacancy concentration, where the vacancies are randomly distributed within the graphene sample. The error bars associated with the MD results in Fig. 10 indicate the range of the obtained fracture stress. Figures 10a and 10b show that the analytical and neural network models agree reasonably with the average value from MD simulations over the temperature range considered.

However, as temperature increases beyond 1200 K, the difference between the analytical and neural network results shows an increasing trend. It should be noted that this trend could be a result of the fact that the neural network had been trained using data up to a temperature of 950 K, but the network was used to predict the fracture stress up to 1500 K to investigate the network's ability to extrapolate. In view of the trends observed in Fig. 10, it would be interesting to train a neural network using MD data instead of the analytical model and examine the resulting trends.

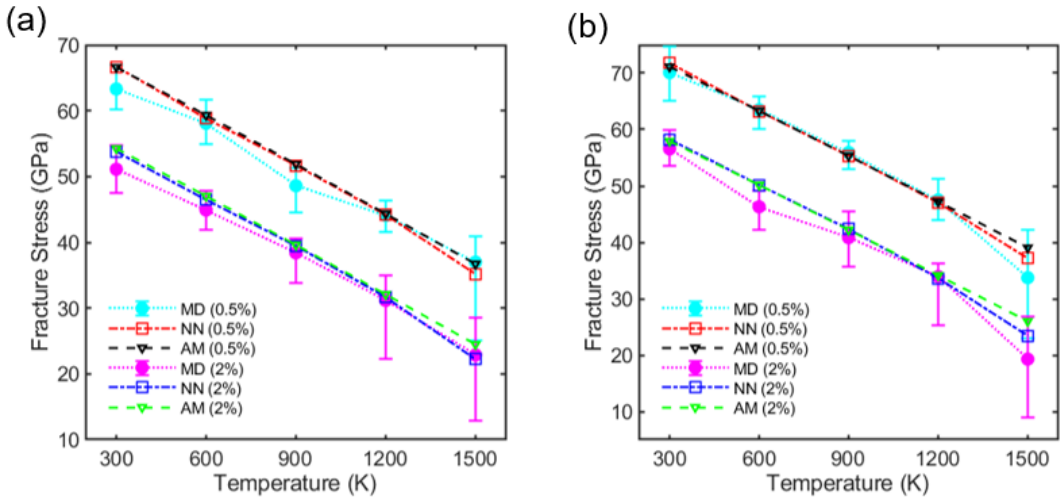

Figure 10 Comparison of results from MD simulations, shallow neural network (NN), and analytical model (AM) for (**a**) armchair and (**b**) zigzag graphene.

3.1.3 *Modelling Neural Networks using MD Data*

We conducted MD simulations of uniaxial tensile tests on 1000 defective graphene samples to obtain data to train neural networks. The vacancy concentrations of the samples were varied from 0.1% to 2% with an increment of 0.1% and the simulation temperature varied from 300 K to 900 K with an increment of 150 K. The loading was applied to samples along the armchair direction. Considering the high computational cost associated with MD simulations, we focused our attention to a fixed strain rate. In the literature, MD simulations of uniaxial tensile tests are generally conducted at strain rates of around 10^9 s^{-1}, and the fracture stress obtained under this strain rate agrees reasonably with the experimental results [9]. Therefore, we used this strain rate of 10^9 s^{-1} for the MD simulations. Ten MD simulations were conducted for each vacancy concentration and

the average fracture stress of the ten samples was used for modelling. There were thus 100 data samples to model the neural network.

A neural network containing three hidden neurons was selected for the modelling, the two inputs being the vacancy concentration and temperature, and the output is the fracture stress. The total number of learnable parameters in the network is 13. Due to the limited number of training samples, we first trained a network (NN_T) while excluding the data pertinent to vacancy concentrations of 0.5% and 2% for independent testing (i.e. 10 samples). The primary objective of training the network NN_T is to predict the influence of temperature on the fracture stress, whereas the vacancy concentration is considered to be the secondary variable when selecting training samples. It should be noted that the network was trained using data up to 1.9%; hence, and the testing with the vacancy concentration of 2%, which is slightly outside the training data, reveals the extrapolation ability of the network (see Fig. 11a). Subsequently, we trained another network (NN_V), while keeping the data pertinent to the samples at temperatures of 450 K and 750 K for independent testing (i.e. 40 samples). The primary objective of training the network NN_V is to predict the influence of vacancy concentration on the fracture stress, whereas the temperature is considered to be the secondary variable when selecting training samples. It can be seen in Fig. 11b that the neural network NN_V was able to predict the influence of vacancy concentration at the two temperatures with a high accuracy. It should be noted that the fracture stress obtained from MD simulation in Fig. 11b shows some fluctuations, which suggests that the distribution of defects could have a higher influence on the fracture stress than the influence of vacancy concentration, when the vacancy concentrations are comparable to each other. In addition, even though the network NN_V has only seen data pertinent to three widely spaced temperatures, it was yet able to identify the nonlinear variation of the fracture stress and generalize it over a large span of temperatures, including intermediate ones.

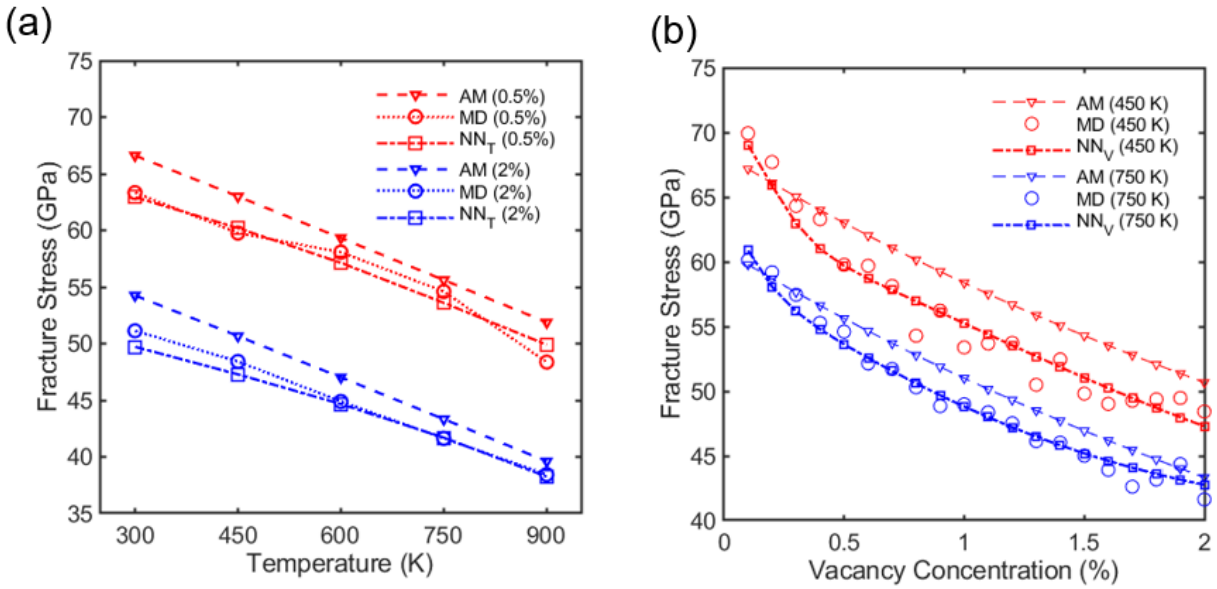

Figure 11 Comparison of the variation of fracture stress obtained from MD simulations, neural network (NN), and analytical model (AM) with (**a**) temperature and (**b**) vacancy concentration.

Even though neural networks are recommended for use within the limits of the data used to train them, it is useful to have an understanding of their extrapolating ability, which could be beneficial in practical applications. In order to further investigate the ability of the two networks NN_v and NN_T to extrapolate fracture stress of graphene samples beyond the range of data that has been used to train them, we employed the networks to predict fracture stresses at two higher vacancy concentrations (2.5% and 3%) and at two higher temperatures (1200 K and 1500 K). Additional MD simulations were also conducted to obtain fracture stresses under similar conditions. Figure 12a compares the MD simulation results with the predictions of the two neural networks at three vacancy concentrations (2%, 2.5%, and 3%). It should be noted that the networks have been trained with data containing vacancy concentrations up to 2% and temperatures up to 900 K. The predictions of NN_T is reasonably accurate even at higher vacancy concentrations, whereas the predictions of NN_V converges to approximately 41 GPa as temperature increases up to 900 K (see Fig. 12a). Similarly, at higher temperatures, predictions of NN_V converges to around 41 GPa as the vacancy concentration increases (see Fig. 12b). On the other hand, the network NN_T demonstrates some ability to extrapolate the influence of vacancy concentration even at higher

temperatures. Given that the two networks have similar architecture, the main difference between them is in the number of training examples, which are 42 and 63 for NN_v and NN_T, respectively. Therefore, the better extrapolating ability of NN_T may be attributed to the higher number of training examples. Also, the effect of vacancy concentration on stress is greater than that of temperature; and higher temperatures significantly change this dependence on vacancy concentration (see Fig. 7(a)), and both these effects will reduce the performance of NN_v under extrapolation. Both networks, however, accurately predict the fracture stress inside the range of their training data (see Fig. 11).

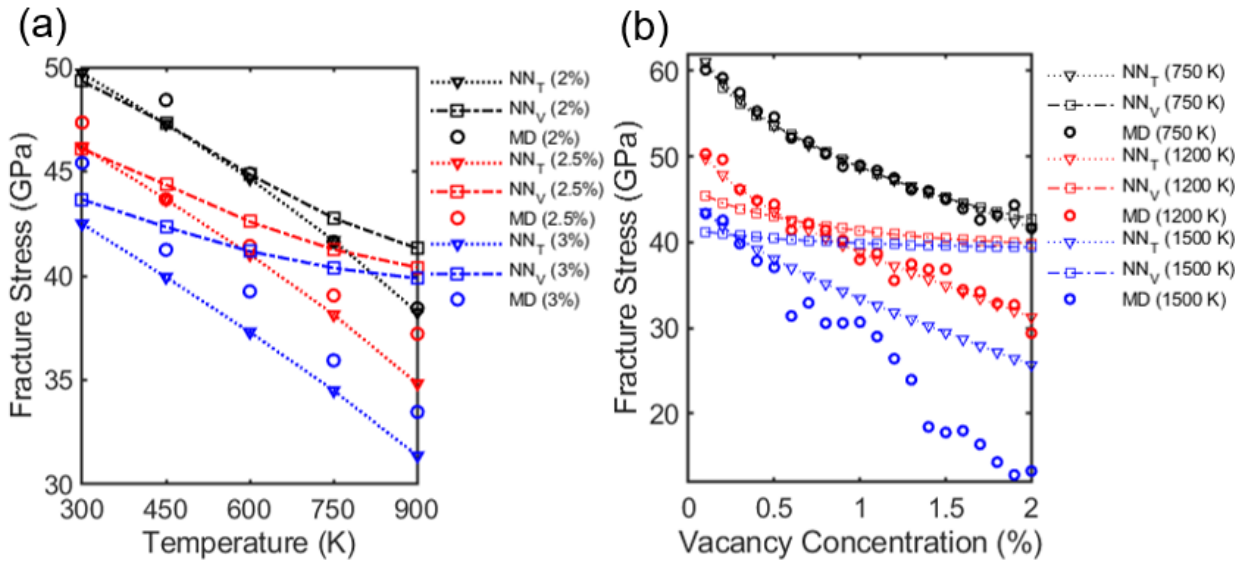

Figure 12 Use of the neural networks NN_T and NN_V for extrapolation (**a**) with respect to vacancy concentration and (**b**) with respect to temperature.

Molecular dynamics simulations demonstrate that the influence of vacancy concentration on the fracture stress at 1500 K is significantly different to that at 750 K and 1200 K (see Fig. 12b). In order to test the performance of neural networks in this nonlinear regime, we trained another network (NN_{TV}) using MD simulation data where the vacancy concentrations of the samples were varied from 0.1% to 2% with an increment of 0.1% and the simulation temperature varied from 300 K to 1500 K with an increment of 300 K (i.e. 100 data samples in total). Architecture of this network is similar to that of NN_T and NN_V. The predictions of NN_{TV} were compared with MD simulation results at temperatures of 750 K and 1350 K. Figure 13 shows that the network captures

the influence of vacancies on the fracture stress with a reasonably high accuracy. These observations suggest that neural networks have strong capability to identify underlying features pertinent to complex nanomechanical problems and could become a useful tool for nanomechanical design. Moreover, as demonstrated in the Section 3.2.2, sensitivity analysis of a trained network could provide novel insights into complex nanomechanical problems.

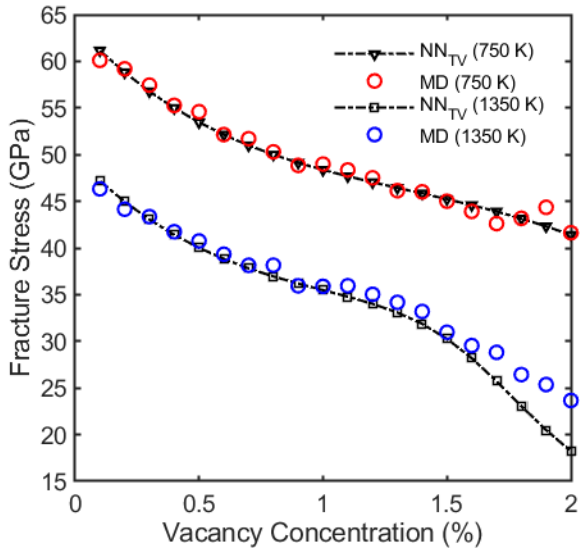

Figure 13 Comparison of the temperature and vacancy concentration dependant fracture stress obtained from MD simulations with the predictions of neural network NN_{TV}.

3.2 *Effects of Defect Distribution*

The developed analytical model (Section 2.2) and the trained neural networks (Sections 3.1.2 and 3.1.3) are unable to predict the influence of defect distribution on the fracture stress of a sample. The stress distribution of a defective sample can be highly complicated and hence challenging to develop analytical solutions for this problem. Moreover, nonlinear stress-strain relations and anisotropic fracture characteristics of graphene make the problem even more challenging. Figure 14 demonstrates the stress distributions of a defective graphene sample at two levels of applied strain along the two principle chiral directions (i.e. armchair and zigzag), where stresses of individual atoms were computed using the definition of virial stress (see Eq. 11). The maximum value of the computed individual atomic stresses (σ_{max}) and the average value of the atomic stress

across the entire sheet (σ_{ave}) are given in Fig. 14.The values of σ_{ave} and σ_{max} of the sample when it is subjected to a strain of 5% along the armchair direction are 43.6 GPa and 84.5 GPa, respectively (see Fig. 14c). However, when the same sample was strained along the zigzag direction up to the same strain, σ_{ave} and σ_{max} of the sample are 39.2 GPa and 69.2 GPa, respectively (see Fig. 14d). Insets of Fig. 14 demonstrate the complex interactions between individual vacancies, which could have a remarkable impact on the crack propagation. Table 3 summarizes the information of the four stress fields shown in Fig 14, which indicates that the maximum stress concentration depends on the both loading direction as well as the applied strain level. In this section, we first develop a shallow network to predict the influence of defect distribution on the fracture stress, which is followed by an effort to employ deep CNNs to tackle this challenging problem.

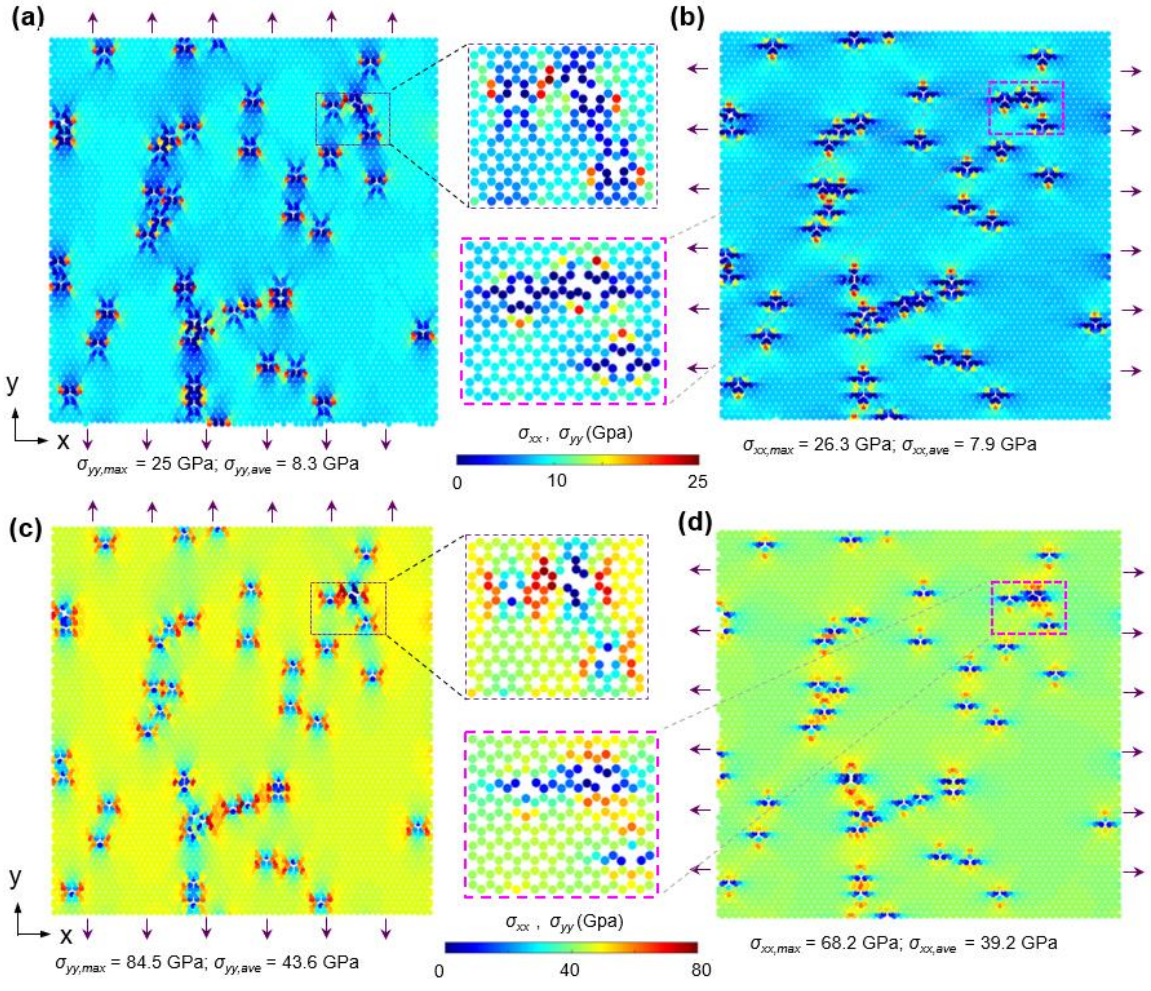

Figure 14 Stress distribution of a graphene sample containing 0.5% of vacancies under an applied strain level of 1% along (**a**) armchair and (**b**) zigzag directions, respectively. The stress state of the

same sample under an applied strain of 5% along the (**c**) armchair and (**d**) zigzag direction. Insets depict the atomic stresses of a selected region.

Table 3 Stress concentration of a graphene sample containing 0.5% of vacancy concentration.

Strain	Loading direction	σ_{max} (GPa)	σ_{avg} (GPa)	$\sigma_{max}/\sigma_{avg}$
1%	armchair	25.0	8.3	3.0
	zigzag	26.3	7.9	3.3
5%	armchair	84.5	43.6	1.9
	zigzag	68.2	39.2	1.7

3.2.1 *Data for Modelling Neural Networks*

We performed nanoscale uniaxial tensile tests of armchair graphene samples containing five levels of vacancy concentrations from 0.1% to 0.5% with an increment of 0.1%. For each vacancy concentration, 250 samples were prepared with different distributions of vacancies, and the simulations performed at a temperature of 300 K under a strain rate of 10^9 s^{-1}. Figure 15 shows the distribution of the obtained fractures stresses. The red horizontal bar in each blue box indicates the median fracture stress, where the bottom and top edges of the blue box represent the 25^{th} and 75^{th} percentiles, respectively. The dashed lines beside the box indicate the limit of the most extreme data points that are not considered outliers. The outliers are denoted by the + symbol, which represents comparatively low fracture stresses due to coalescence of individual vacancies forming a crack like defect. It should be noted that the variation of fracture stresses below the median for each vacancy concentration is always greater than that above the median. This is to be expected because configurations where vacancies are close to each other will lead to a range of lower fracture stresses. When the vacancies are far away from each other however, their interaction is not significant, and the fracture stress is almost independent of the vacancy distribution. In the subsequent section, we use this data to develop a shallow neural network and later, in Section 3.3, we present the development of a deep CNN.

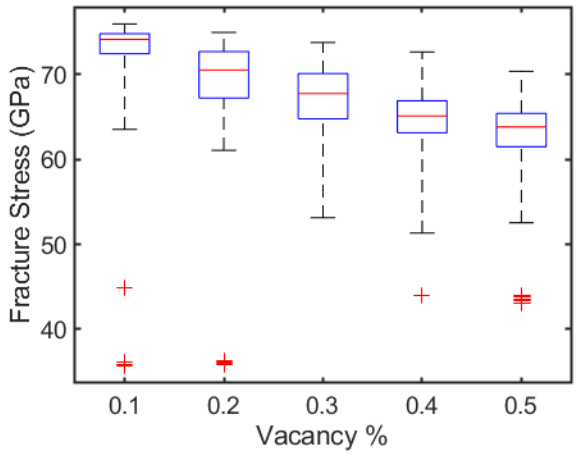

Figure 15 Distribution of the fracture stress obtained from 1250 MD simulations.

3.2.2 *Modelling of a Shallow Neural Network*

Inputs to the network were the five shortest distances between two adjacent vacancy defects and the output was the fracture stress. The five inputs (d_1 to d_5) are arranged in the input vector in such a way that d_1 is the shortest distance between two vacancies in a sample and d_2 is the second shortest distance, etc… Interestingly, three hidden neuron we found to be sufficient for this complex problem. The network (SNN-1) was trained using 926 data samples and a separate 247 samples used for validation, where the distribution of training and validation data are shown in Fig. 15. The outliers (denoted by the + symbol in Fig. 15) were excluded from the data samples. After the training, the network was tested using a new set of data obtained from MD simulations, which comprise of 250 graphene samples containing a vacancy concentration of 0.3%. It can be seen in Fig. 16a that the network is able to capture the underlying trend of the training data; however, the variability of the output is significantly high. Root mean square error (RMSE) of the test data is 2.87 (see Fig. 16b). In addition to RMSE, the slope of the regression line (not shown) between output and target (m) is another indication on the performance of the network [64]. Better models need to have slopes close to unity (i.e. similar to the line of equality). As given in Fig. 16b, the value of m for the current model is 0.35, which indicates that the model significantly deviates from a perfect fit.

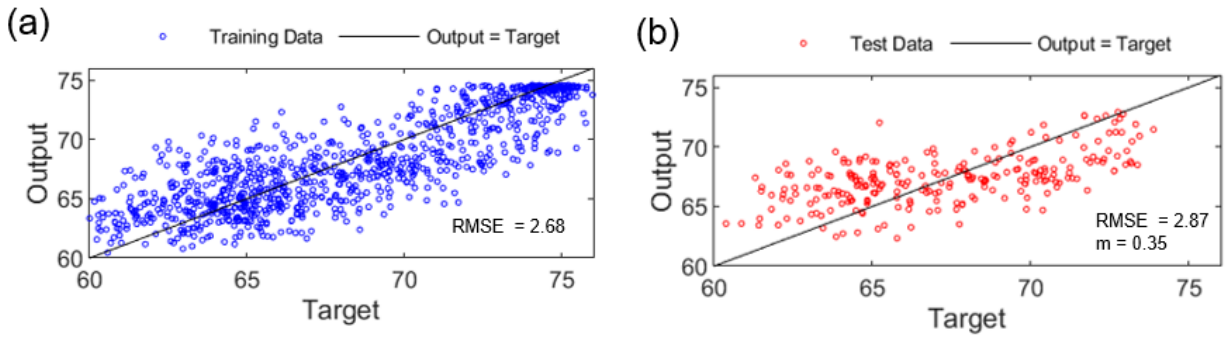

Figure 16 Performance of the shallow neural network in predicting the influence of vacancy distribution on the fracture stress for (a) training set and (b) testing set. The root mean square error (RMSE) of the training and test data, and the slope of the regression line between output and target (m) for the test data are shown on the figures.

Sensitivity analysis of a trained network is a useful way to understand what the network has learnt during the training, and also to obtain an insight into the computations of the network. By following the format of Eq. 1, the fracture stress (σ_f) predicted by the network can be expressed in terms of the five inputs and the associated weights and biases as follows:

$$\sigma_f = f_a^o \left[b_1^{(2)} + \sum_{j=1}^{3} w_{j,1}^{(2)} \, f_a^{(h)} \left(b_j^{(1)} + \sum_{i=1}^{5} w_{i,j}^{(1)} \, d_i \right) \right] \tag{12}$$

where, the weight matrices and bias vectors were found to be,

$$w_{i,j}^{(1)} = \begin{bmatrix} -0.068 & 0.224 & -4.962 \\ 0.795 & -0.935 & -0.284 \\ -1.287 & 1.365 & 0.585 \\ -0.440 & 0.427 & -0.780 \\ -1.499 & 1.617 & -0.395 \end{bmatrix}, \tag{13}$$

$$w_{i,j}^{(2)} = \begin{bmatrix} -5.327 \\ -4.748 \\ -0.365 \end{bmatrix},$$

$$b_j^{(1)} = \begin{bmatrix} -1.784 \\ 1.813 \\ -3.934 \end{bmatrix},$$

$$b_1^{(2)} = -0.057,$$

The range of the data used for training is given in Table 4. It can be noticed that the minimum value of d_1, d_2, and d_3 is 0.14 nm, which is the carbon-carbon bond length in graphene [50]. The similar minimum value of the three variables indicates that at least one sample contains three pairs of vacancies, where the two vacancies in each pair are located next to each other at a distance of a carbon-carbon bond length. Individual vacancies in a graphene sample have to be located very close to each other, approximately within 1 nm, in order to them to interact with each other leading to a significant influence on the fracture stress of the sample [10,11]. The maximum value of d_1 is 3.49 nm (see Table 4), where the two corresponding vacancies are unlikely to interact with each other. However, the presence of a large number of vacancies even at relatively larger distance could have a noticeable influence on the facture stress [11]. It should be noted that the input and output data are transformed linearly during the training by mapping the respective minimum and maximum values of each variable to be -1 and 1, respectively.

Table 4 Ranges of the data used to train the neural network (d_i is in nm and σ_f in GPa).

Variable	d_1	d_2	d_3	d_4	d_5	σ_f
Minimum	0.14	0.14	0.14	0.24	0.28	51.3
Maximum	3.49	4.49	4.54	5.30	5.72	76.0

We conducted a sensitivity analysis of the network by varying one input from 0.14 nm to 3 nm while keeping the other four inputs at 3 nm. Figure 17 shows the variation of the output (i.e. fracture stress) with each input variable. The distances d_1, d_3, and d_5 have the highest influence on the fracture stress. The network has properly learnt that d_1 has the most influence on the fracture stress when its value is less than 0.5 nm, where two vacancies can coalesce to form a crack like defect, leading to a significant reduction in the fracture stress. However, d_1 does not have a noticeable influence on the fracture stress when it is greater than 1 nm. Remarkably, the fracture stress is almost independent of the input d_2 and the effect of d_4 has the next smallest influence on the fracture stress. This could be due to the fact that the effects of d_2 and d_4 are probably accounted for by d_1, d_3 and d_5. Even though, d_1 does not have a noticeable influence on the fracture stress when it is greater than 1 nm, d_3 and d_5 show an influence on the fracture stress in the entire range. This observation could be attributed to the likelihood that the network has learnt in such a way that

the short-range defect interactions are mainly described by d_1, while d_3 and d_5 have been generalized through the entire range.

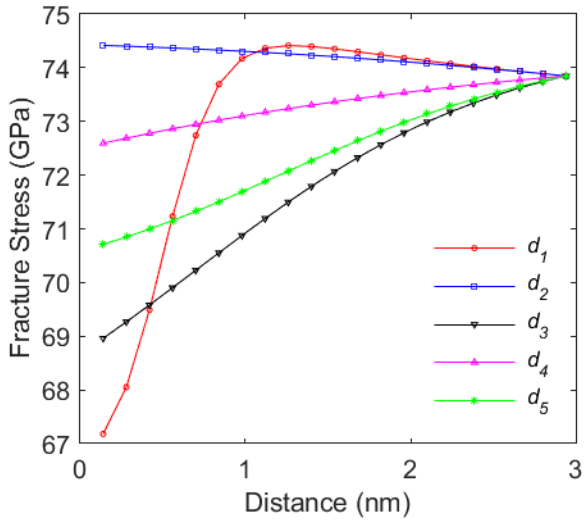

Figure 17 Sensitivity of the network's output to each input variable.

In addition to the distance between two defects, the angle of the line joining the two defects is also an important factor in determining the fracture stress of a sample [2]. Therefore, we trained another shallow network (SNN-2), which uses the sine value of the angle of the line joining two vacancies with the direction normal to the loading direction (sin θ_i; see Fig. 18) as an input, in addition to the distances between the two vacancies. It should be noted that for a given distance, the fracture stress is expected to increase as sin θ_i increases (from 0 to 1) - in the way that the fracture stress increases when distances increase; and therefore, the magnitudes of the two types of input variables (i.e. d_i and sin θ_i) influence the output in a similar direction. This new network receives 10 inputs (5 distances and the 5 corresponding angles) arranged in a vector (d_1, sin θ_1, d_2, sin θ_2, …, d_5, sin θ_5). The optimum number of hidden neurons were found to be 5 and the total number of learnable parameters 61. The additional five inputs slightly reduce the RMSE from 2.87 to 2.7, but the slope of the target-predicted curve increases significantly from 0.35 to 0.52, indicating a better fit than in SNN-1. In the ensuing section, we employ a deep CNN to model the influence of defect distribution on the fracture stress. CNNs are widely used for classification problems, but here we employ CNNs for a regression problem.

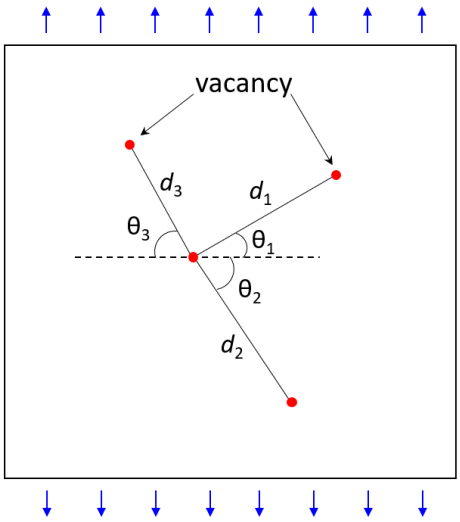

Figure 18 Distance (d_i) and relative angle (θ_i) between vacancies. Blue arrows indicate the loading direction.

3.3 *Modelling of Deep CNN*

In order to model a deep CNN, we reused the 1250 data samples, which were initially used to model the shallow network in Section 3.2.2. An image containing the distribution of vacancies in a sample, where each vacancy is represented by a dot, is selected to be the input for the CNN, and the output is the fracture stress of the sample. In the uniaxial tensile tests, it is noted that the fracture stress is insensitive to a rotation of the testing sample by an angle of 180°. Taking this fact into account, we can have two input images for each value of fracture stress (i.e. the original image and its 180 rotation), which increases the number of examples in the data set up to 2500. Such data augmentation is common in deep learning because large number of training examples are required to train a deep neural network [32,33].

3.3.1 *Modelling Fracture Stress using AlexNet*

First, we performed transfer learning using the pretrained AlexNet, where the last two layers of the AlexNet were replaced by a regression layer [32]. The learning rate of the regression layer was initially kept at 10^{-4} while it was kept at 10^{-8} for all other layers to ensure that the network does not significantly change the already learned weights in the convolutional layers. The learning rate was

dropped by a factor of 2 at every 5 epochs and the training stopped when the validation error did not reach a new minimum in six consecutive iterations. The minibatch size was selected to be 40 and the stochastic gradient descent with 0.9 momentum used for training [33]. After removing the outliers (see Fig. 15), the data set contained 2470 samples, of which 15% was used for testing the network. The remaining 85% was used for training (70%) and validation. The performance of the network is shown on Fig. 19, and it demonstrate that its performance is significantly lower than the performance of the shallow network (see Fig. 16).

It should be noted that this modified AlexNet contains 56,872,321 learnable parameters, whereas the number of training samples is only 1729. This large disparity between the number of learnable parameters and the training examples may be justified considering the fact that the network has been pretrained for image classification using several millions of examples [32]. However, as shown in Fig. 19, the large number of learnable parameters do not allow the network to properly learn the characteristics of the training data. At the same time, the current problem requires AlexNet to extract features in a distribution of defects (represented by a set of dots) and relate them to a unique number in a continuous number range, whereas AlexNet has been originally trained to extract features of 1000 classes of objects (e.g., cars, cats, etc...) and relate them to an integer with a certain confidence level. This slight discrepancy in the feature extraction segment could also contribute to the limited performance of AlexNet in the current problem. Therefore, we created a new CNN from scratch to make sure that the number of learning parameters of the network is comparable with the number of training examples.

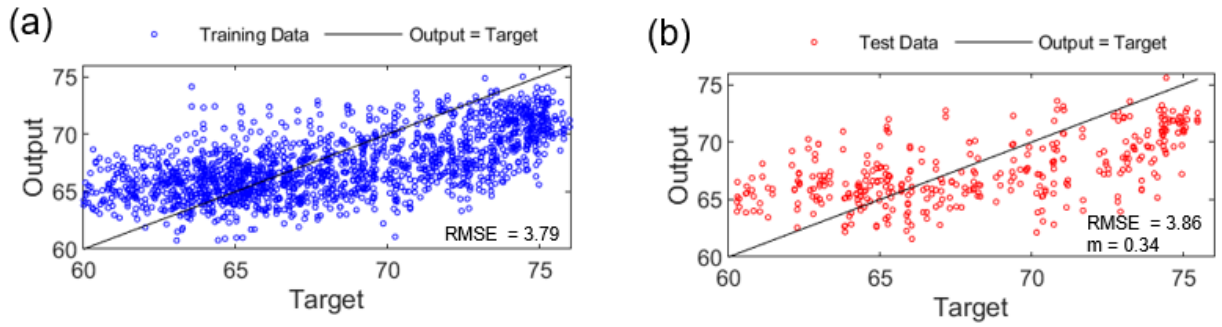

Figure 19 Performance of the AlexNet when it is used for transfer learning. Target-output relationship of data used for (**a**) training and (**b**) testing.

3.3.2 *Modelling a CNN*

Architecture of the developed network (CNN–1) is shown in Fig. 5 and the distribution of learnable parameters within the network given in Table 1. The number of layers and filter sizes were found by trial and error. The number of learnable parameters of the network is 1998 and the total number of training examples 2100. The initial learning rate was selected to be 10^{-4}, and the learning rate decreased by a factor of 2 at every 5 epochs and the training stopped when the validation error did not reach a new minimum in six consecutive iterations. Other parameters associated with training were similar to the those used in the transfer learning (see Section 3.3.1). Performance of the first network is shown in Fig. 20(a), where it can be noticed that the performance of the network on the training set is slightly better than the modified AlexNet (see Fig. 19). However, the network has not been able to properly learn the correlation between the distribution of defects and the fracture stress, which is evident from the large scatter of the target-output plot for the test data (see Fig. 20a).

The performance of the network depends highly on the attributes of the training data. It can be observed in Fig. 15 that the data outside the 25^{th} and 75^{th} percentiles demonstrate a high scatter. Therefore, we trained another network (CNN-2) by taking only the data between the 25^{th} and 75^{th} percentiles for the modelling. It can be seen in Fig. 20(b) that this filtering of data has significantly improved the performance of the network, which can be partly attributed to the fact that the scatter of the data used for the modelling is remarkably less. Interestingly, the network has categorized the training data into five different classes and assigned a specific range of output for each class. The five classes correspond to the five vacancy concentrations considered in the training data (see Fig. 15), and the predicted fracture stress is approximately equal to the average fracture stress of the samples containing a given vacancy concentration.

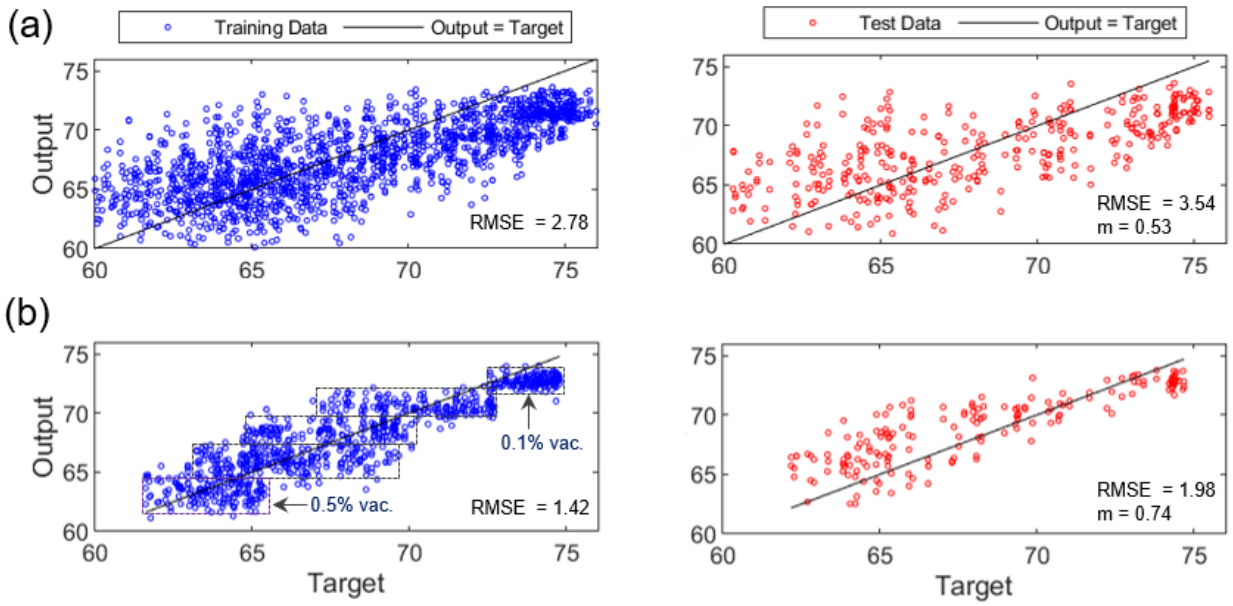

Figure 20 Performance of the developed CNNs: (**a**) trained with all data (**b**) trained with data within the 25th and 75th percentiles.

It should be noted that the input to the network is a large sparse matrix with dimensions of 224×224, which contains a limited information about the distribution of defects. For example, in the case of 0.1% vacancies, the input matrix contains approximately 80 non-zero entries, which is about 0.2% of the entries in the input matrix. Therefore, feature engineering the input could help the network to perform better. In order to explore this hypothesis, we modelled another network (CNN-3) where the input to the network is arranged in a matrix with dimensions of 5×5, which contains information about the distribution of vacancies. The input matrix was prepared as follows: first, the five critically located vacancies are identified by computing the distance between all vacancies; here the vacancies that are located closer to each other are considered to be more critical. Then for each critical vacancy, the five shortest nearest neighbour vacancy distances were calculated, and the values placed in a row of the matrix. The five rows are arranged according the values in first column in such a way that the first entry of the matrix (1,1) is the smallest and therefore the most critical value. The entries of each row increase from left to right. The total number of data samples available for modelling was 625. The network contains three convolutional layers, where each layer is followed by a batch normalization and ReLu layer, and the total number of learnable parameters is 302. Initial learning rate for was selected to be 10^{-3}.

Figure 21 shows that the network demonstrates a slightly better generalization compared to the previous network, suggesting that feature engineering of the input data could contribute to improve the performance of the CNN.

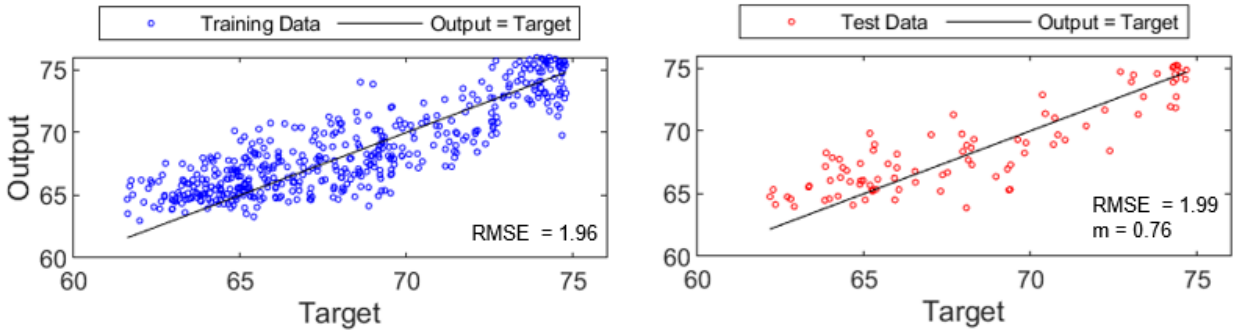

Figure 21 Performance of the CNN modelled using a pre-processed input matrix.

Table 5 compares the architecture and the performance of the two shallow networks and four deep networks. In addition to the RMSE and the slope, *m*, of the test data, the mean absolute error (MAE) is given in the table in order to give an insight on the variation in the errors. Higher difference between RMSE and MAE indicates higher variance in the individual errors in the test data. In the case of shallow networks, introduction of sin (θ) as an input has reduced MAE by 14%, whereas RMSE is reduced by only 6%, which indicates that variance in the individual errors has not reduced significantly. Somewhat similar phenomena can be seen in between AlexNet and CNN-1 as well. The largest improvement in the deep network occurs when the variance in the training data was removed in CNN-2 (through eliminating outliers). It should be noted that CNN-3 shows similar performance with a much less learnable parameters (302 vs 1998). Even though one of the main purposes of deep learning is to minimize human intervention, this result suggests that that significant hand engineering of input data could be helpful for achieving better deep learning models for the problems in the domain of computational materials science.

Table 5 Comparison of neural network models.

Model	Inputs	Learnables	Training samples	RMSE	MAE	m	Remarks
SNN-1	Dist.	22	926	2.87	2.35	0.35	5 inputs
SNN-2	Dist., sin (θ)	61	926	2.70	2.02	0.52	10 inputs
AlexNet	Defect distribution	56,872,321	1,729	3.86	3.14	0.34	224×224 feature matrix
CNN-1	Defect distribution	1,998	2,100	3.54	2.74	0.53	Less learnables
CNN-2	Defect distribution	1,998	1,062	1.98	1.55	0.74	Data in 25^{th} and 75^{th} percentiles
CNN-3	Distances	302	531	1.99	1.57	0.76	5×5 feature matrix; less samples and learnables

4. Conclusions

Our comprehensive neural network study reveals that both shallow and deep networks could serve as computationally efficient tools to predict fracture stress of graphene samples under various processing conditions. Data required to train the neural networks was obtained from a developed analytical solution as well as from molecular dynamics simulations. Our results demonstrate that shallow neural networks have a strong ability to predict the fracture stress of graphene samples and could also possess significant extrapolation capability. In addition, shallow networks are particularly useful when a limited number of training examples are available. On the other hand, deep neural networks require a large number of training examples and can be used to solve highly complicated problems such as the influence of defect distribution in a graphene sample on its fracture stress. Our study suggests that transfer learning could have some inherent limitations when

a deep network, originally trained for a classification problem, is used for a regression problem, even though the feature extraction part of the two cases are somewhat identical. Moreover, significant feature engineering (or pre-processing) of input data may be required to achieve better results when deep learning models are used in the domain of computational materials design.

Acknowledgements

The authors thank NSERC for supporting this research. Computing resources were provided by Compute/Calcul Canada.

References

[1] W.A. Curtin, R.E. Miller, A perspective on atomistic-continuum multiscale modeling, Model. Simul. Mater. Sci. Eng. 25 (2017) 071004. https://doi.org/10.1088/1361-651X/aa8659.
[2] B.A. Moore, E. Rougier, D. O'Malley, G. Srinivasan, A. Hunter, H. Viswanathan, Predictive modeling of dynamic fracture growth in brittle materials with machine learning, Comput. Mater. Sci. 148 (2018) 46–53. https://doi.org/10.1016/j.commatsci.2018.01.056.
[3] A. Hunter, B.A. Moore, M. Mudunuru, V. Chau, R. Tchoua, C. Nyshadham, S. Karra, D. O'Malley, E. Rougier, H. Viswanathan, G. Srinivasan, Reduced-order modeling through machine learning and graph-theoretic approaches for brittle fracture applications, Comput. Mater. Sci. 157 (2019) 87–98. https://doi.org/10.1016/j.commatsci.2018.10.036.
[4] L. Tapaszto, T. Dumitrica, S.J. Kim, P. Nemes-Incze, C. Hwang, L.P. Biro, Breakdown of continuum mechanics for nanometre-wavelength rippling of graphene, Nat Phys. 8 (2012) 739–742.
[5] D.-B. Zhang, E. Akatyeva, T. Dumitrică, Bending Ultrathin Graphene at the Margins of Continuum Mechanics, Phys. Rev. Lett. 106 (2011). https://doi.org/10.1103/PhysRevLett.106.255503.
[6] M.A.N. Dewapriya, S.A. Meguid, R.K.N.D. Rajapakse, Atomistic modelling of crack-inclusion interaction in graphene, Eng. Fract. Mech. 195 (2018) 92–103. https://doi.org/10.1016/j.engfracmech.2018.04.003.
[7] E.B. Tadmor, R.E. Miller, Modeling materials: continuum, atomistic and multiscale techniques, Cambridge University Press, 2011.
[8] M.A.N. Dewapriya, R.K.N.D. Rajapakse, Atomistic and continuum modelling of stress field at an inhomogeneity in graphene, Mater. Des. 160 (2018) 718–730. https://doi.org/10.1016/j.matdes.2018.10.006.
[9] M.A.N. Dewapriya, S.A. Meguid, Atomistic modeling of out-of-plane deformation of a propagating Griffith crack in graphene, Acta Mech. 228 (2017) 3063–3075. https://doi.org/10.1007/s00707-017-1883-7.
[10] M.A.N. Dewapriya, S.A. Meguid, Tailoring fracture strength of graphene, Comput. Mater. Sci. 141 (2018) 114–121. https://doi.org/10.1016/j.commatsci.2017.09.005.
[11] M.A.N. Dewapriya, S.A. Meguid, Atomistic simulations of nanoscale crack-vacancy interaction in graphene, Carbon. 125 (2017) 113–131. https://doi.org/10.1016/j.carbon.2017.09.015.
[12] T. Kirchdoerfer, M. Ortiz, Data-driven computational mechanics, Comput. Methods Appl. Mech. Eng. 304 (2016) 81–101. https://doi.org/10.1016/j.cma.2016.02.001.
[13] T. Kirchdoerfer, M. Ortiz, Data Driven Computing with noisy material data sets, Comput. Methods Appl. Mech. Eng. 326 (2017) 622–641. https://doi.org/10.1016/j.cma.2017.07.039.

[14] C.M. Bishop, Pattern recognition and machine learning, Springer, New York, 2006.
[15] P.V. Balachandran, Machine learning guided design of functional materials with targeted properties, Comput. Mater. Sci. 164 (2019) 82–90. https://doi.org/10.1016/j.commatsci.2019.03.057.
[16] S. Wu, Y. Kondo, M. Kakimoto, B. Yang, H. Yamada, I. Kuwajima, G. Lambard, K. Hongo, Y. Xu, J. Shiomi, C. Schick, J. Morikawa, R. Yoshida, Machine-learning-assisted discovery of polymers with high thermal conductivity using a molecular design algorithm, Npj Comput. Mater. 5 (2019) 66. https://doi.org/10.1038/s41524-019-0203-2.
[17] G. Srinivasan, J.D. Hyman, D.A. Osthus, B.A. Moore, D. O'Malley, S. Karra, E. Rougier, A.A. Hagberg, A. Hunter, H.S. Viswanathan, Quantifying Topological Uncertainty in Fractured Systems using Graph Theory and Machine Learning, Sci. Rep. 8 (2018). https://doi.org/10.1038/s41598-018-30117-1.
[18] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature. 521 (2015) 436–444. https://doi.org/10.1038/nature14539.
[19] A. Mardt, L. Pasquali, H. Wu, F. Noé, VAMPnets for deep learning of molecular kinetics, Nat. Commun. 9 (2018) 5. https://doi.org/10.1038/s41467-017-02388-1.
[20] A. Oishi, G. Yagawa, Computational mechanics enhanced by deep learning, Comput. Methods Appl. Mech. Eng. 327 (2017) 327–351. https://doi.org/10.1016/j.cma.2017.08.040.
[21] B. Lusch, J.N. Kutz, S.L. Brunton, Deep learning for universal linear embeddings of nonlinear dynamics, Nat. Commun. 9 (2018) 4950. https://doi.org/10.1038/s41467-018-07210-0.
[22] D. Finol, Y. Lu, V. Mahadevan, A. Srivastava, Deep convolutional neural networks for eigenvalue problems in mechanics, Int. J. Numer. Methods Eng. 118 (2019) 258–275. https://doi.org/10.1002/nme.6012.
[23] R. Kondo, S. Yamakawa, Y. Masuoka, S. Tajima, R. Asahi, Microstructure recognition using convolutional neural networks for prediction of ionic conductivity in ceramics, Acta Mater. 141 (2017) 29–38. https://doi.org/10.1016/j.actamat.2017.09.004.
[24] Z. Yang, Y.C. Yabansu, R. Al-Bahrani, W. Liao, A.N. Choudhary, S.R. Kalidindi, A. Agrawal, Deep learning approaches for mining structure-property linkages in high contrast composites from simulation datasets, Comput. Mater. Sci. 151 (2018) 278–287. https://doi.org/10.1016/j.commatsci.2018.05.014.
[25] R. Cang, H. Li, H. Yao, Y. Jiao, Y. Ren, Improving direct physical properties prediction of heterogeneous materials from imaging data via convolutional neural network and a morphology-aware generative model, Comput. Mater. Sci. 150 (2018) 212–221. https://doi.org/10.1016/j.commatsci.2018.03.074.
[26] Z. Zhang, Y. Hong, B. Hou, Z. Zhang, M. Negahban, J. Zhang, Accelerated discoveries of mechanical properties of graphene using machine learning and high-throughput computation, Carbon. 148 (2019) 115–123. https://doi.org/10.1016/j.carbon.2019.03.046.
[27] H. Yang, Z. Zhang, J. Zhang, X.C. Zeng, Machine learning and artificial neural network prediction of interfacial thermal resistance between graphene and hexagonal boron nitride, Nanoscale. 10 (2018) 19092–19099. https://doi.org/10.1039/C8NR05703F.
[28] N. Artrith, A. Urban, An implementation of artificial neural-network potentials for atomistic materials simulations: Performance for TiO 2, Comput. Mater. Sci. 114 (2016) 135–150. https://doi.org/10.1016/j.commatsci.2015.11.047.
[29] N. Lubbers, J.S. Smith, K. Barros, Hierarchical modeling of molecular energies using a deep neural network, J. Chem. Phys. 148 (2018) 241715. https://doi.org/10.1063/1.5011181.
[30] J.S. Smith, B.T. Nebgen, R. Zubatyuk, N. Lubbers, C. Devereux, K. Barros, S. Tretiak, O. Isayev, A.E. Roitberg, Approaching coupled cluster accuracy with a general-purpose neural network potential through transfer learning, Nat. Commun. 10 (2019) 2903. https://doi.org/10.1038/s41467-019-10827-4.
[31] S.S. Haykin, Neural networks: a comprehensive foundation, 2nd ed, Prentice Hall, Upper Saddle River, N.J, 1999.

[32] A. Krizhevsky, I. Sutskever, G.E. Hinton, ImageNet Classification with Deep Convolutional Neural Networks, in: Adv. Neural Inf. Process. Syst. 25, 2012: pp. 1097–1105.
[33] I. Goodfellow, Y. Bengio, A. Courville, Deep learning, The MIT Press, Cambridge, Massachusetts, 2016.
[34] K. Simonyan, A. Zisserman, Very Deep Convolutional Networks for Large-Scale Image Recognition, ArXiv14091556 Cs. (2015). http://arxiv.org/abs/1409.1556 (accessed December 6, 2019).
[35] K. Hornik, M. Stinchcombe, H. White, Multilayer feedforward networks are universal approximators, Neural Netw. 2 (1989) 359–366. https://doi.org/10.1016/0893-6080(89)90020-8.
[36] K.-I. Funahashi, On the approximate realization of continuous mappings by neural networks, Neural Netw. 2 (1989) 183–192. https://doi.org/10.1016/0893-6080(89)90003-8.
[37] D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning representations by back-propagating errors, Nature. 323 (1986) 533–536. https://doi.org/10.1038/323533a0.
[38] M.T. Hagan, M.B. Menhaj, Training feedforward networks with the Marquardt algorithm, IEEE Trans. Neural Netw. 5 (1994) 989–993. https://doi.org/10.1109/72.329697.
[39] Y. Lecun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE. 86 (1998) 2278–2324. https://doi.org/10.1109/5.726791.
[40] K. Weiss, T.M. Khoshgoftaar, D. Wang, A survey of transfer learning, J. Big Data. 3 (2016) 9. https://doi.org/10.1186/s40537-016-0043-6.
[41] K. He, X. Zhang, S. Ren, J. Sun, Deep Residual Learning for Image Recognition, in: 2016 IEEE Conf. Comput. Vis. Pattern Recognit. CVPR, IEEE, Las Vegas, NV, USA, 2016: pp. 770–778. https://doi.org/10.1109/CVPR.2016.90.
[42] M.A.N. Dewapriya, R.K.N.D. Rajapakse, Molecular Dynamics Simulations and Continuum Modeling of Temperature and Strain Rate Dependent Fracture Strength of Graphene With Vacancy Defects, J. Appl. Mech.-Trans. Asme. 81 (2014). https://doi.org/10.1115/1.4027681.
[43] M.A.N. Dewapriya, R.K.N.D. Rajapakse, N. Nigam, Influence of hydrogen functionalization on the fracture strength of graphene and the interfacial properties of graphene–polymer nanocomposite, Carbon. 93 (2015) 830–842. https://doi.org/10.1016/j.carbon.2015.05.101.
[44] S. Arrhenius, On the reaction rate of the inversion of non-refined sugar upon souring, Z Phys Chem. 4 (1889) 226–248.
[45] J. Bailey, An attempt to correlate some tensile strength measurements on glass: III, Glass Ind. 20 (1939) 95–99.
[46] A.D. Freed, A.I. Leonov, The Bailey criterion: Statistical derivation and applications to interpretations of durability tests and chemical kinetics, Z. Angew. Math. Phys. 53 (2002) 160–166. https://doi.org/10.1007/s00033-002-8148-5.
[47] Y.I. Jhon, Y.M. Jhon, G.Y. Yeom, M.S. Jhon, Orientation dependence of the fracture behavior of graphene, Carbon. 66 (2014) 619–628. https://doi.org/10.1016/j.carbon.2013.09.051.
[48] M. Abramowitz, I.A. Stegun, Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables, ninth Dover printing, tenth GPO printing, Dover, New York, 1964.
[49] S. Plimpton, Fast parallel algorithms for short-range molecular dynamics, J Comput Phys. 117 (1995) 1–19. https://doi.org/10.1006/jcph.1995.1039.
[50] S.J. Stuart, A.B. Tutein, J.A. Harrison, A reactive potential for hydrocarbons with intermolecular interactions, J. Appl. Phys. 112 (2000) 6472–6486. https://doi.org/DOI:10.1063/1.481208.
[51] K.G.S. Dilrukshi, M.A.N. Dewapriya, U.G.A. Puswewala, Size dependency and potential field influence on deriving mechanical properties of carbon nanotubes using molecular dynamics, Theor. Appl. Mech. Lett. 5 (2015) 167–172. http://dx.doi.org/10.1016/j.taml.2015.05.005.
[52] M.A.N. Dewapriya, A.S. Phani, R.K.N.D. Rajapakse, Influence of temperature and free edges on the mechanical properties of graphene, Model. Simul. Mater. Sci. Eng. 21 (2013) 065017.
[53] F. Cleri, S.R. Phillpot, D. Wolf, S. Yip, Atomistic Simulations of Materials Fracture and the Link between Atomic and Continuum Length Scales, J. Am. Ceram. Soc. 81 (1998) 501–516. https://doi.org/10.1111/j.1151-2916.1998.tb02368.x.

[54] L.Q. Xu, N. Wei, Y.P. Zheng, Mechanical properties of highly defective graphene: from brittle rupture to ductile fracture, Nanotechnology. 24 (2013) 7. https://doi.org/10.1088/0957-4484/24/50/505703.
[55] C. Carpenter, D. Maroudas, A. Ramasubramaniam, Mechanical properties of irradiated single-layer graphene, Appl. Phys. Lett. 103 (2013) 013102. https://doi.org/10.1063/1.4813010.
[56] M.A.N. Dewapriya, S.A. Meguid, Comprehensive molecular dynamics studies of the ballistic resistance of multilayer graphene-polymer composite, Comput. Mater. Sci. 170 (2019) 109171. https://doi.org/10.1016/j.commatsci.2019.109171.
[57] D. Hossain, M.A. Tschopp, D.K. Ward, J.L. Bouvard, P. Wang, M.F. Horstemeyer, Molecular dynamics simulations of deformation mechanisms of amorphous polyethylene, Polymer. 51 (2010) 6071–6083. https://doi.org/10.1016/j.polymer.2010.10.009.
[58] A.P. Thompson, S.J. Plimpton, W. Mattson, General formulation of pressure and stress tensor for arbitrary many-body interaction potentials under periodic boundary conditions, J. Chem. Phys. 131 (2009) 154107. https://doi.org/10.1063/1.3245303.
[59] T. Ohta, Controlling the Electronic Structure of Bilayer Graphene, Science. 313 (2006) 951–954. https://doi.org/10.1126/science.1130681.
[60] K.S. Novoselov, V.I. Falko, L. Colombo, P.R. Gellert, M.G. Schwab, K. Kim, A roadmap for graphene, Nature. 490 (2012) 192–200.
[61] M.A. Rafiee, J. Rafiee, I. Srivastava, Z. Wang, H. Song, Z.-Z. Yu, N. Koratkar, Fracture and Fatigue in Graphene Nanocomposites, Small. 6 (2010) 179–183. https://doi.org/10.1002/smll.200901480.
[62] C. Lee, X. Wei, J.W. Kysar, J. Hone, Measurement of the Elastic Properties and Intrinsic Strength of Monolayer Graphene, Science. 321 (2008) 385–388. https://doi.org/10.1126/science.1157996.
[63] P. Zhang, L. Ma, F. Fan, Z. Zeng, C. Peng, P.E. Loya, Z. Liu, Y. Gong, J. Zhang, X. Zhang, others, Fracture toughness of graphene, Nat. Commun. 5 (2014).
[64] M.A.N. Dewapriya, W.P.S. Dias, T.S.G. Peiris, Modelling of Tsunami Intensity Using Artificial Neural Network and Fuzzy Inference Systems, Int. J. Sci. Eng. Res. 8 (2017) 1–7.