Appl. Math. Mech. -Engl. Ed., **44**(7), 1085–1100 (2023)

APPLIED MATHEMATICS AND MECHANICS (ENGLISH EDITION)

https://doi.org/10.1007/s10483-023-2991-9

Peri-Net-Pro: the neural processes with quantified uncertainty for crack patterns*

M. KIM[1], G. LIN[2,3,†]

1. Samsung Electronics, Yongin-si, Gyeonggi-do 17113, Republic of Korea;
2. Department of Mathematics, Purdue University, West Lafayette, IN 47907-2067, U. S. A.;
3. School of Mechanical Engineering, Purdue University, West Lafayette, IN 47907-2067, U. S. A.

(Received Nov. 1, 2022 / Revised Mar. 16, 2023)

Abstract This paper develops a deep learning tool based on neural processes (NPs) called the Peri-Net-Pro, to predict the crack patterns in a moving disk and classifies them according to the classification modes with quantified uncertainties. In particular, image classification and regression studies are conducted by means of convolutional neural networks (CNNs) and NPs. First, the amount and quality of the data are enhanced by using peridynamics to theoretically compensate for the problems of the finite element method (FEM) in generating crack pattern images. Second, case studies are conducted with the prototype microelastic brittle (PMB), linear peridynamic solid (LPS), and viscoelastic solid (VES) models obtained by using the peridynamic theory. The case studies are performed to classify the images by using CNNs and determine the suitability of the PMB, LBS, and VES models. Finally, a regression analysis is performed on the crack pattern images with NPs to predict the crack patterns. The regression analysis results confirm that the variance decreases when the number of epochs increases by using the NPs. The training results gradually improve, and the variance ranges decrease to less than 0.035. The main finding of this study is that the NPs enable accurate predictions, even with missing or insufficient training data. The results demonstrate that if the context points are set to the 10th, 100th, 300th, and 784th, the training information is deliberately omitted for the context points of the 10th, 100th, and 300th, and the predictions are different when the context points are significantly lower. However, the comparison of the results of the 100th and 784th context points shows that the predicted results are similar because of the Gaussian processes in the NPs. Therefore, if the NPs are employed for training, the missing information of the training data can be supplemented to predict the results.

Key words neural process (NP), peridynamics, crack pattern, molecular dynamic (MD) simulation, machine learning, Gaussian process regression, convolutional neural network (CNN)

* Citation: KIM, M. and LIN, G. Peri-Net-Pro: the neural processes with quantified uncertainty for crack patterns. *Applied Mathematics and Mechanics* (*English Edition*), **44**(7), 1085–1100 (2023) https://doi.org/10.1007/s10483-023-2991-9

† Corresponding author, E-mail: guanglin@purdue.edu

Project supported by the National Science Foundation of U. S. A. (Nos. DMS-1555072, DMS-2053746, and DMS-2134209), the Brookhaven National Laboratory of U. S. A. (No. 382247), and U. S. Department of Energy (DOE) Office of Science Advanced Scientific Computing Research Program (Nos. DE-SC0021142 and DE-SC0023161)

Chinese Library Classification O313.4, O346.1
2010 Mathematics Subject Classification 74A25, 74M20, 74R10, 68T05

1 Introduction

A major problem in solid mechanics is associated with the construction of discontinuities in areas that do not exist. Typical examples of discontinuity formation include the failure of a mechanical component due to cracking and breakage caused by a dynamic load, high-temperature gradient, shock, or explosion; damage to people's possessions due to human negligence or mistakes in daily life; automobile accidents that can cause cracks on the car's body; crack formation on the blades of thermoelectric and wind power plants due to the effects of high temperature or wind; and damage to nuclear reactors caused by nuclear fission. Research on the causes of damage to motile objects is actively underway. Many studies on fracture mechanics utilize the finite element method (FEM). However, the FEM has major problems in fracture mechanics calculations. First, the FEM is calculated based on partial differential equations, but there are no partial derivatives on the crack surface. In addition, only the approximate solution can be obtained in the FEM calculations. The result is dependent on the mesh, and user errors can affect the results. Peridynamics is employed to compensate for these problems. The peridynamic theory was first introduced by Seleson et al.[1], Silling et al.[2], and Silling[3]. By applying integral equations directly to the crack surface, peridynamics can compensate for the absence of partial derivatives, improve the accuracy, and avoid the problems caused by the FEM. From this perspective, peridynamics is the most appropriate theory for studying cracks or damage. The peridynamic theory is applied not only to cracks but also to various materials that develop cracks, such as membranes and nanofibers[4], composites, brittle materials[5], and viscoelastic materials[6]. The molecular dynamic (MD) simulation tools, such as LAMMPS[7] and PERIDIGM[8], contain packages that use the peridynamic theory for simulations. These packages have been extensively used in crack simulation research, especially on various material properties. For instance, the prototype microelastic brittle (PMB)[9], linear peridynamic solid (LPS)[10] , and viscoelastic solid (VES)[11] models are included in MD simulation tools. However, MD simulation tools require increased computational time as their size increases. One way to reduce the computational cost is to apply deep learning methods. In recent years, convolutional neural networks (CNNs) have exhibited improved performance and accuracy compared with classical fully connected networks, particularly for highly structured data applications such as peridynamic models[12]. CNNs have the advantage of pre-training, learning the datasets generated from highly accurate models based on the peridynamic theory, and encoding the process of the model into a neural network. Such neural networks can provide immediate results through training and testing processes. The novel contributions of the present study are three-fold.

(i) Neural processes (NPs)[13] are employed to predict and quantify the uncertainties in crack patterns. NPs combine the advantages of a neural network and a Gaussian process with the data obtained from the peridynamic theory. We demonstrate through numerical results that NPs can significantly improve the computational efficiency during training and evaluation, resulting in faster and more accurate results. The numerical experiments demonstrate that the predicted results match the ground truth.

(ii) We demonstrate that NPs can quantify the uncertainties in predicting crack patterns. Through the training process, the variance is significantly reduced as the number of epochs increases, which indicates that the predicted results gradually match the ground truth.

(iii) Finally, the CNNs and NPs are applied based on the data obtained from the peridynamic theory. Applying NPs can increase the calculation speed using the trained data and reduce the computational cost.

The remainder of this paper is organized as follows. In Section 2, the peridynamic models

(PMB, LPS, and VES) are briefly introduced. This study employs the NPs, incorporating a Gaussian process into the neural network to enable predictions despite the loss of input data information. Section 3 introduces the NP algorithm. In Section 4, the method to obtain the input data for training by using the three peridynamic models is explained. Finally, in Section 5, the CNNs and NPs are trained and evaluated with the data obtained from the three peridynamic models. In particular, the NPs are employed to infer the value of the variance as the number of epochs increases, and the prediction accuracy is evaluated.

2 The peridynamic models (PMB, LPS, and VES)

In peridynamics, all points in the material are connected by bonds. Figure 1 shows the composition of the peridynamic model. Suppose that there is a body that contains a circle. There is an $\boldsymbol{x}$ at the center of the circle whose radius is denoted as δ (horizon). All the points within the circle are connected, and are referred to as the family of $\boldsymbol{x}$. Therefore, peridynamics is the continuation of classical continuum mechanics. The most crucial point of peridynamics is that it uses an integral equation. In contrast, the FEM uses a partial differential equation for the crack analysis. However, there are no partial derivatives on the crack surface. Peridynamics can solve this problem by applying integral equations directly to the crack surface. From this perspective, peridynamics is the most appropriate theory for studying cracks or damage. The equation of motion based on the peridynamic model is as follows:

$$\rho(\boldsymbol{x})\ddot{u}(\boldsymbol{x},t) = \int_{\mathcal{F}_{\boldsymbol{x}}} T(\boldsymbol{u}(\boldsymbol{x}',t) - \boldsymbol{u}(\boldsymbol{x},t), \boldsymbol{x}' - \boldsymbol{x})\mathrm{d}V_{\boldsymbol{x}'} + b(\boldsymbol{x},t), \quad t \geqslant 0. \tag{1}$$

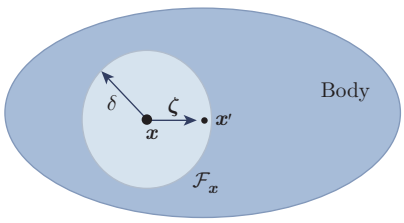


Fig. 1 Composition of the peridynamic model (color online)

The left term represents the equation of motion, where ρ represents the mass density, and $\ddot{u}$ represents the acceleration. On the right, $\mathcal{F}_{\boldsymbol{x}}$ is the family of $\boldsymbol{x}$, T is the pairwise force function of the bond which connects the points $\boldsymbol{x}$ and $\boldsymbol{x}'$, $\boldsymbol{u}$ is the displacement of the point $\boldsymbol{x}$ according to the time domain, and b represents the external body force density. The following definitions of $\boldsymbol{\zeta}$ and $\boldsymbol{\eta}$ are required for the conservation of linear momentum and conservation of angular momentum: $\boldsymbol{\zeta}$ is the position vector in the initial arrangement, and $\boldsymbol{\eta}$ is the relative displacement vector.

$$\boldsymbol{\zeta} = \boldsymbol{x}' - \boldsymbol{x}, \tag{2}$$

$$\boldsymbol{\eta} = \boldsymbol{u}' - \boldsymbol{u}. \tag{3}$$

$\boldsymbol{\zeta} + \boldsymbol{\eta}$ indicates the current relative position vector between the points. The T function follows the two properties,

$$T(-\boldsymbol{\eta}, -\boldsymbol{\zeta}) = -T(\boldsymbol{\eta}, \boldsymbol{\zeta}) \quad \forall \boldsymbol{\eta}, \boldsymbol{\zeta}, \tag{4}$$

$$(\boldsymbol{\zeta} + \boldsymbol{\eta}) \times T(\boldsymbol{\eta}, \boldsymbol{\zeta}) = 0 \quad \forall \boldsymbol{\eta}, \boldsymbol{\zeta}. \tag{5}$$

Equations (4) and (5) represent the conservation of linear momentum and the conservation of angular momentum, respectively. The conditions of T satisfy both the conservation of linear momentum and the conservation of angular momentum. In this study, the crack patterns are formed by applying the above-mentioned peridynamic theory. The PMB, LPS, and VES models are applied in the case study of crack patterns according to the material types. Here, we briefly provide the physical interpretations and explain the differences among the PMB, LPS, and VES models. The PMB model has a fixed Poisson's ratio, while the LPS model can deal with different Poisson's ratios. The LPS model is state-based, while the PMB model is bond-based. The PMB model evolves from isotropic and micro-elastic models. The strength of the bond is dependent only on the bond stretch s. The equation is expressed as

$$y = \|\boldsymbol{\zeta} + \boldsymbol{\eta}\|, \tag{6}$$

$$s = \frac{\|\boldsymbol{\zeta} + \boldsymbol{\eta}\| - \|\boldsymbol{\zeta}\|}{\|\boldsymbol{\zeta}\|} = \frac{y - \|\boldsymbol{\zeta}\|}{\|\boldsymbol{\zeta}\|}, \tag{7}$$

where s is the bond stretch and consists of the current relative position vector between the points and the position vector in the initial arrangement. The PMB model is defined as follows:

$$F(y(t), \boldsymbol{\zeta}) = g(s(t, \boldsymbol{\zeta}))\mu(t, \boldsymbol{\zeta}), \tag{8}$$

$$g(s) = cs \quad \forall s, \tag{9}$$

$$\mu(t, \boldsymbol{\zeta}) = \begin{cases} 1, & \text{if} \quad s(t', \boldsymbol{\zeta}) < s_0 \quad \text{for all} \quad 0 \leqslant t' \leqslant t, \\ 0, & \text{otherwise.} \end{cases} \tag{10}$$

The function F is a product of the g and μ functions. The g function is a linear scalar-valued function consisting of c and s, which are the spring constant and the bond stretch, respectively. The μ function, a component of the F function, is a function of the time and the position vector. It has the values of 0 and 1, depending on the conditions in Eq. (10). If the bond stretch is less than the critical bond stretch s_0, μ is 1. Otherwise, $\mu = 0$. The spring constant c and the critical bond s_0 are expressed as follows:

$$c = \frac{18K}{\pi\delta^4}, \tag{11}$$

$$s_0 = \sqrt{\frac{10G_0}{\pi c\delta^5}} = \sqrt{\frac{5G_0}{9K\delta}}, \tag{12}$$

$$G_0 = \frac{\pi c s_0^2 \delta^5}{10}. \tag{13}$$

The spring constant c is only appropriate for three-dimensional models, and consists of the bulk modulus K and δ (horizon boundary in the peridynamic model). The critical bond stretch s_0 is a function of the shear modulus G_0 and the horizon δ. The shear modulus G_0 consists of the spring constant c, the critical bond stretch s_0, and the horizon δ in the boundary. The scalar force state of the PMB model is expressed as follows:

$$t_{\mathrm{PMB}} = \frac{1}{2}\frac{18K}{\pi\delta^4}\frac{\|\boldsymbol{\eta} + \boldsymbol{\zeta}\| - \|\boldsymbol{\zeta}\|}{\|\boldsymbol{\zeta}\|}. \tag{14}$$

The elastic properties of the LPS and VES models are defined by the bulk modulus and the shear modulus in conjunction with the horizon boundary. However, the VES model requires additional relaxation parameters and time constants. The scalar force state of the LPS and

VES models can be described as follows. For the LPS model,

$$t_{\mathrm{LPS}} = -\frac{3K\theta}{m}\omega x + \alpha\omega e^{\mathrm{d}}, \tag{15}$$

where m, θ, and e^{d} are the weighted volume, the dilatation, and the deviatoric extension state, respectively. The bulk modulus is K, and the shear modulus G has a related term[2]

$$\alpha = \frac{15G}{m}.$$

For the VES model,

$$t_{\mathrm{VES}} = -\frac{3K\theta}{m}\omega x + (\alpha_\infty + \alpha_i)e^{\mathrm{d}} - \alpha_i\omega e^{\mathrm{db}(i)}. \tag{16}$$

For the VES model, the key constituent is the decomposition of the scalar extension state into the dilatation and deviatoric parts, as well as the additive decomposition of the deviatoric extension state into the elastic e^{d} and back extension $e^{\mathrm{db}(i)}$ states. $\alpha = \alpha_\infty + \alpha_i$, and $0 < \alpha_i < \frac{15\mu}{m}$[11, 14].

3 The NPs

In this section, we introduce the NPs. Neural networks are nonlinear functions. Gaussian processes[15] provide the stochastic framework for learning the distribution of a wide range of nonlinear functions. Thus, with limited data, Gaussian processes can capture the probabilistic nature and uncertainty. However, neural networks are more computationally scalable than Gaussian processes because of their ability to handle vast amounts of data. Therefore, the NP model combines the advantages of neural networks and Gaussian processes to compensate for each other's disadvantages. The NPs are examined through the diagram shown in Fig. 2. The massive flow of the NPs proceeds as follows. Information flows from the context points through the potential space z to the new predictions with the target points. z is a random variable that turns the NPs into a probabilistic model, and captures the uncertainty of the function. Once the model is trained, the posterior distribution of z can be used before making the predictions.

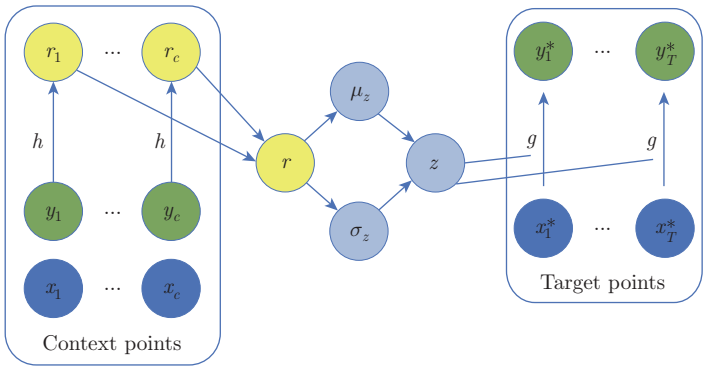


Fig. 2 Diagram of the NPs (color online)

In summary, the given data can be largely divided into context points and target points, and y_T^* can be predicted by using the given context and target points.

The step-by-step procedure of the NPs is listed as follows.

(i) The context points pass through the neural networks h to produce their respective representations $r_1, r_2, \cdots, r_c$.

(ii) All the representations $r_1, r_2, \cdots, r_c$ from each context point are aggregated to obtain a single r, which is then used to parameterize the distribution of z.

$$p(z|x_{1:c}, y_{1:c}) = N(\mu_z(r), \sigma_z^2(r)). \tag{17}$$

(iii) To obtain the prediction point y_T^*, z is sampled, and the result is concatenated with the target point x_T^* to obtain the predictive distribution of y_T^* by passing it through the neural network decoder g.

The NPs are inferred in the variational inference framework. By inferring the approximate posterior $q(z|c_t, t)$, we can evaluate the prediction $p(y_T^*|z,\ x_T^*)$. The approximate posterior $q(z|c_t, t)$ is chosen for the prediction point y_T^* to use the neural network encoder h. The same h is used to map the context and target points to obtain the aggregated r, which in turn is mapped to μ_z and σ_z. Finally, the parameters of the encoder h and the decoder g are trained by the evidence of the lower bound (ELBO),

$$\varphi_{\text{ELBO}} = E_{q(z|c_t,t)}\Big(\sum_{t=1}^{T} \ln p(y_T^*|z, x_T^*) + \ln \frac{q(z|c_t)}{q(z|c_t, t)}\Big), \tag{18}$$

where c_t denotes context.

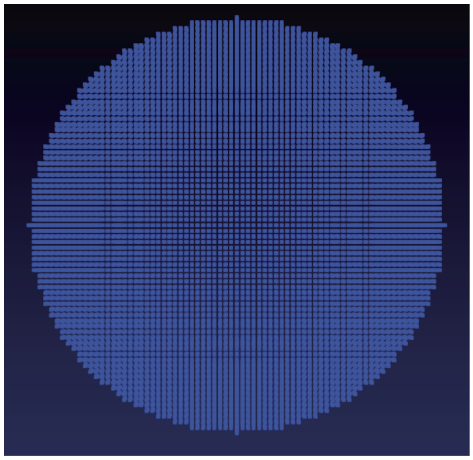
The first term in brackets is the expected log-likelihood of the target set. It is evaluated by using the first sampling $z \sim q(z|c_t, t)$. The second term of the ELBO has a normalization effect, which is a negative Kullback-Leibler (KL) divergence between $q(z|c_t)$ and $q(z|c_t, t)$. This term indicates that a summary of the contexts is close to the summary of the targets.

4 Preparation of the data by using the peridynamic theory and models

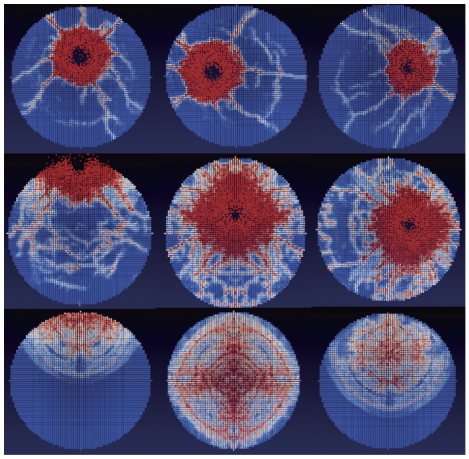
In this study, peridynamics is used to obtain the crack patterns of a moving disk. LAMMPS is used for the simulations. The PMB, VES, and LPS models are used in the case study of the peridynamic model to apply deep machine learning. The crack patterns are penetrated by a spherical indenter perpendicular to the disk moving in the x-, y-, and z-directions. The data are obtained by changing four parameters. (i) The hitting points are set by increasing and decreasing the x- and y-coordinates around the center of the disk. The disk is penetrated by the indenter evenly throughout the cylindrical disk. (ii) The velocity parameters are changed to 100 m/s and 100.1 m/s with a slight difference. The reason for this is to determine how to classify the subtle differences in the velocity when classifying the modes by deep machine learning. (iii) The radii of the indenter are adjusted to 0.007 m and 0.008 m, respectively. (iv) To obtain the crack pattern of the moving disk, not that of the static disk, we change the moving directions of the disk.

The cylindrical disk consists of 103 110 particles with a radius of 0.037 m and a thickness of 0.002 5 m. Each particle has a volume fraction of $V_i = 1.25 \times 10^{-10}\,\text{m}^3$. The density of the disk material is $\rho = 2\,200\,\text{kg/m}^3$. Thus, three modified parameters are applied to the moving disk, which is generated by the PMB, VES, and LPS models for the case study. First, to apply the PMB model, the "peri/pbs" pair style is used. The pair constants for the simulations are as follows: $c = 1.686\,3 \times 10^{22}$, $\delta = 0.001\,500\,1$, $s_{00} = 0.000\,5$, and $\alpha = 0.25$, where c is the spring constant for peridynamic bonds, the horizon δ is the cutoff distance for non-local interactions, and the constants s_{00} and α correspond to material-dependent parameters for the critical bond stretch. Second, we use the "peri/lps" pair style for the LPS model, which consists of the bulk modulus K, the shear modulus G, the horizon δ, s_{00}, and α. The bulk modulus (14.9×10^9), the shear modulus (14.9×10^9), the horizon δ (0.001 500 1), s_{00} (0.000 5), and α (0.25) are used to generate the crack patterns of the LPS model. Finally, to set up the VES model, the "peri/ves" pair style, consisting of the bulk modulus K, the shear modulus G, the horizon δ, s_{00}, α, m_λ, and m_τ, is used. m_λ and m_τ represent the viscoelastic relaxation parameter and

the time constant, respectively. We generate crack patterns for the VES model by applying the bulk modulus $K = 14.9 \times 10^9$, the shear modulus $G = 14.9 \times 10^9$, $\delta = 0.001\,500\,1$, $s_{00} = 0.000\,5$, $\alpha = 0.25$, $m_\lambda = 0.5$, and $m_\tau = 0.001$. The data-set is created for deep machine learning, and the models and simulation settings are used in the case studies. We calculate the simulations for 1 000 time-steps in one simulation, and store the output as a dump file, assuming that 1 000 time-steps are meaningful by tuning the simulation step-by-step for the case model to obtain more sophisticated crack patterns. The pizza.py toolkit[16] is used to change the EnSight data format to visualize the crack patterns by using the dump files, and is then applied to the PARAVIEW[17–18] tool to visualize the crack patterns. Figure 3(a) shows the initial disk model before the crack patterns are formed. In Fig. 3(b), the first, second, and third rows show the crack patterns generated by the PMB, LPS, and VES models, respectively.



(a) Initial disk



(b) Crack patterns

Fig. 3 Initial disk and crack patterns generated by the LPS, PMB, and VES models (color online)

5 Method and results

This study uses supervised machine learning with CNNs to classify the crack patterns in cases according to the PMB, LPS, and VES models and two-dimensional (2D) regression problems with the NPs. We train the deep machine learning methods by using MD simulation data, and

investigate how to accurately predict the crack pattern by using these deep machine learning methods. In this section, we study how to define the data-set to perform supervised machine learning, how to label the training, validation, and test the data-sets, and how to set up the structures in the CNNs and NPs. After training, validate, and testing, the loss function is calculated to evaluate the performance of the training and validation tests. We then proceed with the case studies according to the success rate and accuracy of the predicted results.

5.1 Preparation of the data for classification

For the classification problem, the MATLAB toolbox Matconvnet[19] is used. First, the crack patterns are stored as dump files through the MD simulation based on the peridynamic theory. These dump files are converted into the 64×64 numerical python (NumPy) array and saved as the data. The three types of data are the training, validation, and testing data-sets. The total number of dump files is 10 800, and 12 classification modes with different radii and velocities of the indenter and different moving directions of the disk are used for the image classification. The 12 classification modes are listed in Table 1. We randomly shuffle all the data to reduce the data imbalance, divide the number of data according to each sample, and assign them to the training, validation, and test data-sets. In addition, 30% of the training data is designated as a validation data-set to observe the overfitting and evaluate the training. In these processes, we use 6 300 training data-sets, 2 700 validation data-sets, and 1 800 test data-sets. In the NPs, the PyTorch[20] deep learning implementation library is used for training and testing. The crack data-set has the same format as the MNIST[21] data. Through the MD simulations using the peridynamic theory, 10 800 dump files are obtained as the output. These are converted to the NumPy array and 28×28 portable network graphic files. The training data-set, the training label, the test data-set, and the test label are stored separately and then converted into the ubyte.gz file to obtain the data and label similar to the MNIST dataset. Through these processes, 9 000 training data, 9 000 training labels, 1 080 test data, and 1 080 test labels are obtained.

Table 1 Twelve modes for image classification

Classification mode	Three variables		
	r/m	$v/(\mathrm{m}\cdot\mathrm{s}^{-1})$	Moving direction
1	0.007	100	x
2	0.007	100.1	x
3	0.007	100	y
4	0.007	100.1	y
5	0.007	100	z
6	0.007	100.1	z
7	0.008	100	x
8	0.008	100.1	x
9	0.008	100	y
10	0.008	100.1	y
11	0.008	100	z
12	0.008	100.1	z

5.2 Composition of data-sets for the CNNs and the NPs

For the CNNs, the training and test data-sets are converted into a .mat file for application in Matconvnet. The .mat file consists of "images" and "meta" formats. First, the "images" part consists of "labels", "set", and "data". The "labels" are used to configure the labels for randomly mixed data. In the "labels", the data-set is randomly labeled from 1 to 12 to classify the 12 modes. The "set" specifies the training, test, and validation data-sets. For example, 1, 2, and 3 can be referred to as the training data, the test data, and the validation data, respectively. The "data" part stores the crack pattern images with a NumPy array in the form of a four-dimensional (4D) single. For instance, if the 4D single has a size of $64 \times 64 \times 1 \times 10\,800$,

there are 10 800 grey images with 64×64 pixels. The "meta" structure contains the "set" and "classes". The "set" is related to the "set" in "images", where 1, 2, and 3 are the training data, the test data, and the validation data, respectively. The "classes" are specified from 1 to 12, and are related to the data modes we want to classify. In addition, our data-set is made similar to the MNIST data type to apply the NPs. All the data are compressed in ubyte.gz in the forms of training sets, training labels, test sets, and test labels. The data are randomly mixed before storing the data and labels. Because the validation data-set has yet to be set up, the validation data-set can be specified to use a portion of the entire training set. The labels range from 1 to 12, the same as the number of classifications. The data-set is arranged such that they could be easily seen at a glance by attaching each 28×28 pixel image side-by-side to resemble the MNIST data type. All other data types are ndarrays, an N-dimensional array.

5.3 CNNs for classification

Deep learning has become a powerful machine learning tool that has made it possible to perform various cognitions and inferences by utilizing the human knowledge stored in annotated data. CNNs are widely used to solve computer vision problems such as image classification and object recognition[22–24]. These neural networks consist of several layers of neurons between the input and the output, which can be shared with neurons in other layers to facilitate the communication of information. Each connection unit acts as a linear or nonlinear operator defined by a set of parameters. We use the Alexnet[25] structure to classify the modes of crack patterns. Two processes can be used to classify the annotated data. The first is feature extraction. This process takes the gray 2D image of $a \times a$ pixels as the input. The input image forms a $1 \times c$ feature vector through each layer of neurons. The second process is feature classification. This process classifies the feature vector into the desired 12 classification modes. These processes can be explained in greater detail. First, the feature extraction process requires three layers. The first is the convolutional layer, which receives the input image of size $W_1 \times H_1 \times D_1$, and generates an output image of size $W_2 \times H_2 \times D_2$. Four hyperparameters are required in this process, i.e., the horizontal and vertical spatial size F, the stride S, the zero-padding P, and the number of filters N_f. The input image creates an output image according to the relation among the four hyperparameters. The relation equations are expressed as follows:

$$W_2 = \frac{W_1 E + 2P}{S} + 1, \tag{19}$$

$$H_2 = \frac{H_1 E + 2P}{S} + 1, \tag{20}$$

$$D_2 = N_\mathrm{f}. \tag{21}$$

The volume of the output image is determined through these equations. The calculations are illustrated in Fig. 4. When an input image of $64 \times 64 \times 1$ passes through the convolutional layer by using the above-mentioned relation equations with the following hyperparameters: 11×11 filter size, 100 filters, stride (1), and zero padding (0), the output images of $18 \times 18 \times 100$ are obtained. The second layer is the Max pooling layer[25] that receives the input image passed through the convolutional layer and downsamples it. Therefore, it reduces the computational load and memory usage, which speeds up the calculation. We set the Max pooling to 3 in this study. By applying the Max pooling layer to $18 \times 18 \times 100$ images obtained through the convolutional layer, the input images are converted to $6 \times 6 \times 100$. The ReLU layer[25] is an activation function applied to the neuron at each pixel location. By changing all values that are less than 0 to 0, the computational speed is significantly increased. The ReLU activation function is defined as follows:

$$f_{\mathrm{ReLU}}(x) = \max(0, x), \tag{22}$$

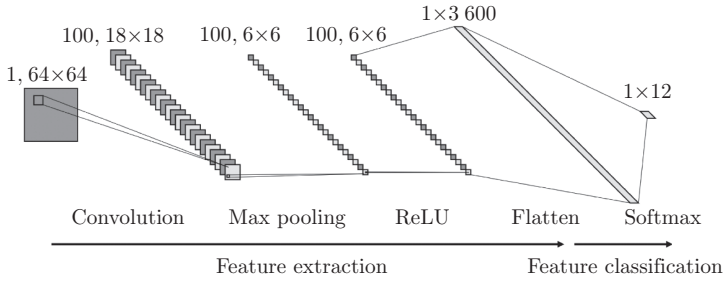


Fig. 4 Architecture of CNNs for classification of 12 modes

where x is the pixel intensity that controls the amount of information passing through the network. Thus, we examine the feature extraction process. When we look at the feature classification process, the output images of $6 \times 6 \times 100$ obtained through the three layers mentioned above are arranged in the form of $1 \times 3\,600$ vectors through the flattening process. This is because our ultimate goal is to classify this into 12 classification modes. Thus, the flattened $1 \times 3\,600$ vectors are classified as 1×12 through the Softmax layer[26]. The Softmax layer is often utilized in the final layer of a neural-network-based classifier. It takes a vector with arbitrary real-valued score (in z), and reduces it to a vector with values between 0 and 1. The process is expressed as follows:

$$z^{[L]} = w^{[L]} a^{[L-1]} + b^{[L]}. \tag{23}$$

To compute z, we need the Softmax activation function

$$t = \mathrm{e}^{z^{[L]}} \tag{24}$$

and the $z^{[L]} = (12, 1)$ dimensional vector related to the classification for 12 modes.

$$a^{[L]} = \frac{\mathrm{e}^{z^{[L]}}}{\sum_{i=1}^{12} t_i}. \tag{25}$$

The output a is going to be the vector t but will be normalized to the sum of 1.

5.4 Results of training and testing by using CNNs with success rates

In this section, we examine the results of training, validation, and testing and the success rates in classifying modes by using specified data and CNNs. As previously mentioned, for the 9 000 training data-sets, 30% of the training data are used as the validation data-sets in the training process. Figure 5 shows the training and validation results according to the PMB, LPS, and VES models. During the training, the hyperparameters are set, such as the learning rate of 0.001, the batch size of 50, and the number of epochs of 100. In Fig. 5, we observe the objective value and determine whether the training is appropriate or not. The blue line represents the training error, while the red line indicates the validation error. The objective value is the same as that in the error graph, and some errors occur at the start of training. However, as the number of epochs increases, it gradually converges to zero. It is also possible to evaluate the overfitting of the training through Fig. 5. If overfitting occurs during the training process, the validation line may not converge to 0, or the value could decrease and converge to a certain value. In this case, we can simplify the model, or reduce overfitting by regularization. In this study, as shown in Fig. 5, the training and validation errors converge to zero without overfitting or underfitting. Finally, we test the 1 200 images based on the training data, evaluate the accuracy of sample predictions, and classify the test images based on the success rates.

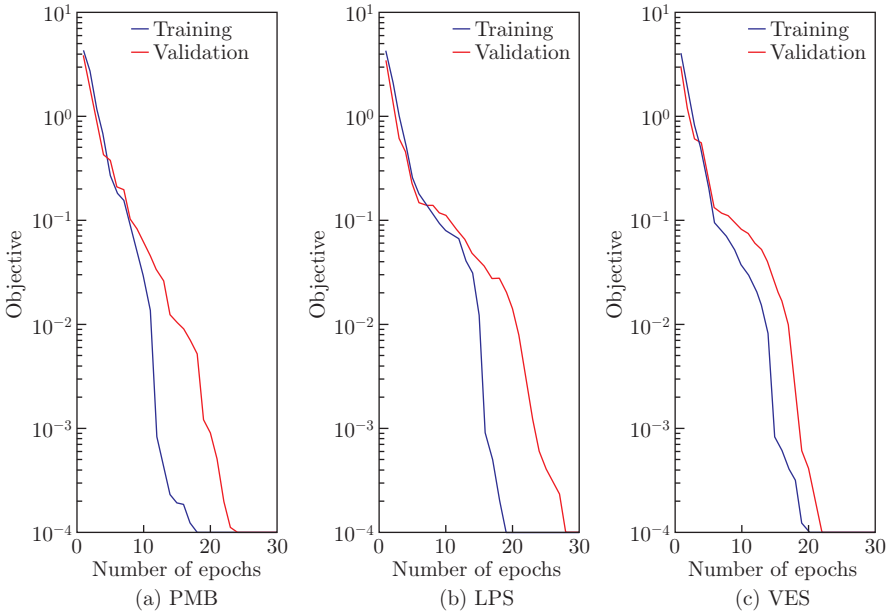


Fig. 5 Objective plots for the PMB, LPS and VES models (color online)

Table 2 summarizes the classification results according to the models using the training data. The test image number and label (mode) indicate the image data tested and the associated labels, respectively. The CNN prediction results (mode) show the predictions of the test images.

Table 2 CNN prediction results compared with the test image data

Model	Test image number: label (mode)	CNN prediction result (mode)
PMB	1:4 (mode)	4 (mode)
	2:11 (mode)	11 (mode)
	3:5 (mode)	5 (mode)
	4:3 (mode)	3 (mode)
	5:7 (mode)	7 (mode)
	6:2 (mode)	2 (mode)
	7:4 (mode)	4 (mode)
	8:8 (mode)	8 (mode)
LPS	Test image 1:10 (mode)	10 (mode)
	Test image 2:7 (mode)	7 (mode)
	Test image 3:3 (mode)	3 (mode)
	Test image 4:5 (mode)	5 (mode)
	Test image 5:2 (mode)	2 (mode)
	Test image 6:6 (mode)	6 (mode)
	Test image 7:1 (mode)	1 (mode)
	Test image 8:12 (mode)	12 (mode)
VES	Test image 1:5 (mode)	5 (mode)
	Test image 2:8 (mode)	8 (mode)
	Test image 3:3 (mode)	3 (mode)
	Test image 4:10 (mode)	10 (mode)
	Test image 5:9 (mode)	9 (mode)
	Test image 6:7 (mode)	7 (mode)
	Test image 7:5 (mode)	5 (mode)
	Test image 8:3 (mode)	3 (mode)

The predictions of the test results according to the model are successful. We obtain the success rate, and the relation equation is expressed as follows:

$$\frac{\text{The number of the correct test images}}{\text{The total number of the test images}} \times 100\%. \tag{26}$$

The prediction results of the image classification into 12 modes with the training data have success rates between 99.6% and 99.8%.

5.5 2D regression problem results by using the NPs

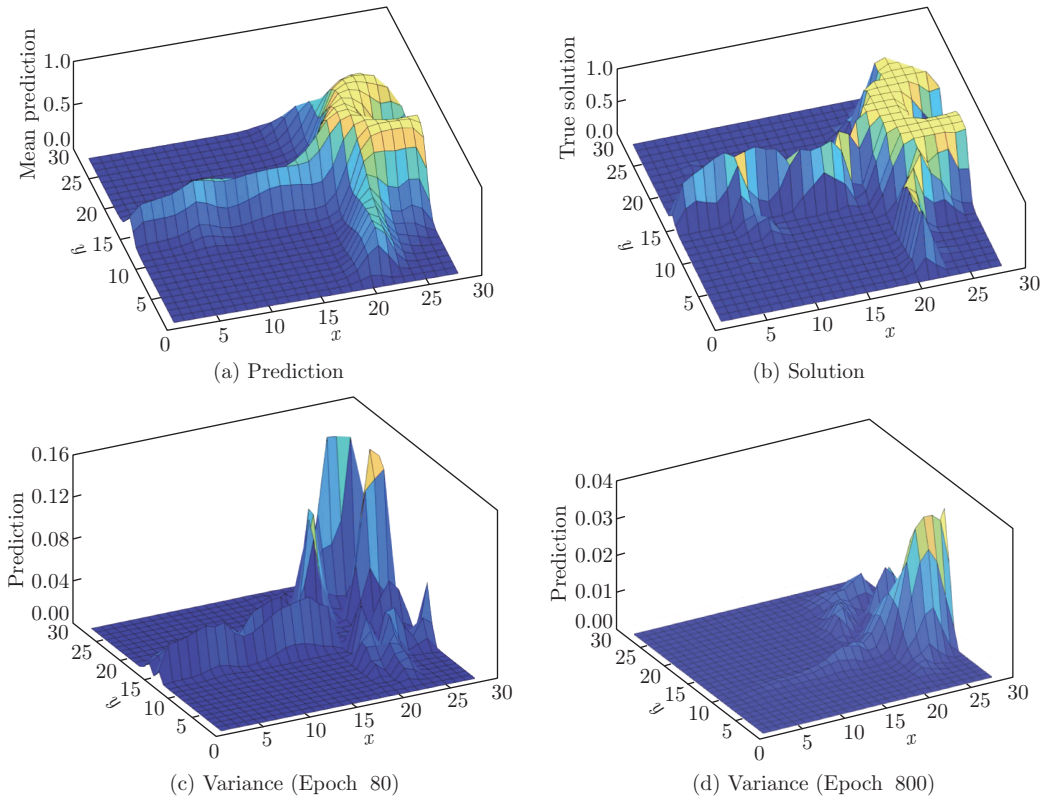
So far, we have classified the crack patterns according to 12 classification modes by using CNNs. In this section, we attempt to predict the crack patterns by solving the 2D regression problem with the NPs. As previously mentioned, 9 000 training data and 1 800 test data are used. The data structure is similar to that of the MNIST data. The size of each image is 28×28 pixels. The NPs are used to solve the 2D regression problem. The experimental set-up of the NPs is as follows. The hyperparameters are set as follows: the input batch size of 128 for training and the number of epochs of 3 000. The dimension of representation r is 128 to obtain each representation through the neural networks from the context points. This is associated with the linear layers of the neural networks that yield 128 representations through three linear layers (3×400, 400×400, and 400×128). In the neural networks, we use the ReLU layer as an activation function and increase the training efficiency. The representation obtained through the neural networks is used to parametrize the distribution of z, which is related to Eq. (17). Through this process, the dimension of the global latent variable z is also 128.

Using the obtained z and target points, we predict the crack patterns through g of the linear layers. The linear layers of g are 130×400, 400×400, 400×400, and 400×1. As a result, the crack patterns can be predicted. The loss function is the binary cross-entropy. The relation equation is expressed as follows:

$$H_P(q) = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot \lg p + (1 - y_i) \cdot \lg(1 - p). \tag{27}$$

Considering the classification modes, we assume that Classification modes 1 and 2 exist. y represents the label for Classification mode 1, and p is the predicted probability that a point is Classification mode 1 among all N points. This indicates that for each classification mode 1 point, it adds $\lg p$ to the loss, that is, the log probability of it being Classification mode 1. On the contrary, it adds $\lg(1 - p)$, that is, the log probability of it being Classification mode 2, for each classification mode 1 point. The model is trained by using standard adaptive moment (ADAM) estimation optimization algorithms, and backpropagation is implemented in PyTorch. The learning speed of the ADAM optimization is set to 0.001.

The network predictions are evaluated by using the above network settings. The results are shown in Fig. 6. The network prediction and true solution can be found at the top of the line, and are quite similar upon the comparison of the results. In the next line, we present the results of the variance obtained from the NPs as the number of epochs increases. As a result of the variance in Fig. 6, the variance significantly decreases when the prediction process proceeds from Epoch 1 to Epoch 800, while the range is 0–0.035. The results show that the predicted results match the true solution, while the variance value decreases as the epoch iterations increase. The variance can be used as an indicator of the accuracy of the network prediction. In addition, the location with the largest variance is where additional samples should be considered. This can be used for the experimental design to determine additional optimal sample locations. Using the structure of the generated NPs, the predicted results obtained from the LPS model through training and testing are shown in Fig. 7. The figure shows the results of the predictions of the test data. In the training process, the context points are set to the 10th, 100th, 300th, and 784th, which represent the test results accordingly in 3 000 epochs. The first figure shows



(a) Prediction (b) Solution

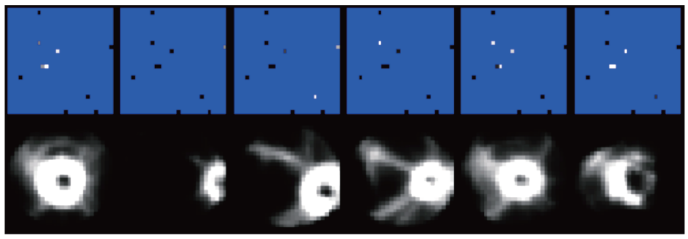
(c) Variance (Epoch 80) (d) Variance (Epoch 800)

Fig. 6 Comparison of the prediction of the NPs and the true solution for the crack pattern problem. The prediction and solution in the two figures on the top row are obtained from the test data-set and are not considered by the NPs during the training process. The prediction and true solution are represented in the top figures, while the bottom figures show the variance at different numbers of the epochs (color online)

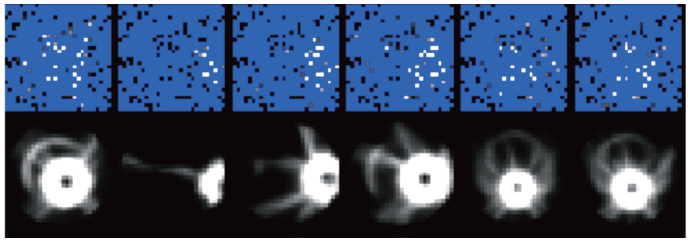
the results of the 3 000 epochs with the 10th context points. It can be seen from Fig. 7(a) that the test images are not accurately predicted because the information of the test images is insufficient to predict the data in the training process by setting the 10th context points. The second figure shows the results of the test images predicted with the 100th context points. In this process, the crack pattern is blurred, but can be predicted gradually. The last image presents the results of the test images predicted after the final training is completed at epoch 3 000 with the 784th context points. It can be seen that the crack patterns are accurately predicted. It is even more surprising that with the 100th and 300th context points, the crack patterns are accurately predicted along with the results with the context points of the 784th. Compared with classical neural networks, the NPs enable accurate predictions even when using data with insufficient information.

6 Conclusions

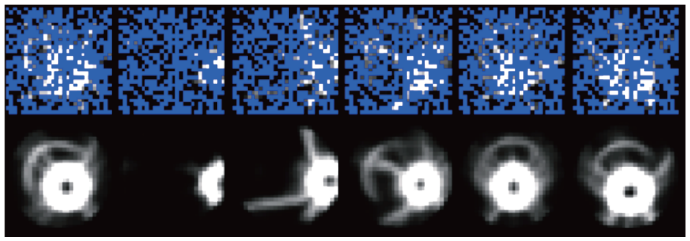
We use peridynamics to obtain the crack patterns by changing four variables, i.e., the indenter velocity, the radius of the indenter, the hitting location of the disk, and the moving direction of the disk. Peridynamics can compensate for the disadvantages of the FEM and generate more accurate crack patterns. In addition, the PMB, LPS, and VES models complement for insufficient data in case studies with CNNs by using peridynamics. First, the modes of the crack



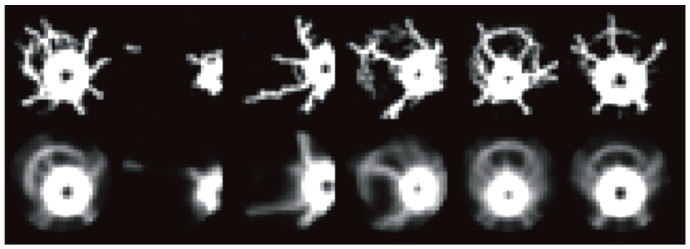
(a) Prediction results with 10th context points



(b) Prediction results with 100th context points



(c) Prediction results with 300th context points



(d) Prediction results with 784th context points

Fig. 7 2D regression problem results for crack patterns with 10th, 100th, 300th, and 784th context points at the number of epochs of 3 000 (color online)

pattern are classified, and the case study of models is conducted with CNNs. The data obtained through peridynamics are labeled according to each classification mode, and supervised learning is performed. By evenly distributing the data, the problem of imbalanced data is reduced. A training data-set of 30% is designated as the validation data-set to deal with the overfitting problem in the training process. Training is optimized by tuning the hyperparameters to reduce overfitting. As a result, a 99.6%–99.8% success rate is achieved, and the modes are accurately classified.

Finally, the NPs are used to solve the 2D regression problems. These NPs are optimized to address the regression problems by combining the advantages of neural networks and Gaussian processes. In particular, the data obtained through peridynamics are similar to the MNIST data, and are classified into the training and test data-sets for training. We also set the context points to the 10th, 100th, 300th, and 784th to obtain the results of the test images. When the

number of epochs is low, the prediction is not good enough, regardless of the context points. However, as the number of epochs increases, the results improve. In the final epoch 3 000, even in the case of the 100th context points, the prediction is sufficiently accurate as that in the case of the 784th context points. In other words, the NPs enable the accurate predictions of crack patterns by using data with limited information.

Conflict of interest The authors declare no conflict of interest.

References

[1] SELESON, P., PARKS, M. L., GUNZBURGER, M., and LEHOUCQ, R. B. Peridynamics as an upscaling of molecular dynamics. *Multiscale Modeling & Simulation*, **8**(1), 204–227 (2009)

[2] SILLING, S., EPTON, A., WECKNER, M., XU, O., and ASKARI, J. Peridynamic states and constitutive modeling. *Journal of Elasticity*, **88**(2), 151–184 (2007)

[3] SILLING, S. Reformulation of elasticity theory for discontinuities and long-range forces. *Journal of the Mechanics and Physics of Solids*, **48**(1), 175–209 (2000)

[4] BOBARU, F., SILLING, S. A., and JIANG, H. Peridynamic fracture and damage modeling of membranes and nanofiber networks. 11*th International Conference on Fracture*, Taylor & Francis, London (2005)

[5] ASKARI, E., XU, J., and SILLING, S. Peridynamic analysis of damage and failure in composites. 44*th AIAA Aerospace Sciences Meeting and Exhibit*, American Institute of Aeronautics and Astronautics, AIAA 2006-88, Reston (2006)

[6] NIKABDULLAH, N., AZIZI, M. A., ALEBRAHIM, R., and SINGH, S. S. K. The application of peridynamic method on prediction of viscoelastic materials behaviour. *AIP Conference Proceedings*, **1602**(1), 357–363 (2014)

[7] PLIMPTON, S. Fast parallel algorithms for short-range molecular dynamics. *Journal of Computational Physics*, **117**(1), 1–19 (1995)

[8] PARKS, M. L., LITTLEWOOD, D. J., MITCHELL, J. A., and SILLING, S. A. *Peridigm User Guide*, Tech. Report SAND2012-7800, Sandia National Laboratories, Albuquerque (2012)

[9] SILLING, S. A. and ASKARI, E. A meshfree method based on the peridynamic model of solid mechanics. *Computers & Structures*, **83**, 1526–1535 (2005)

[10] PARKS, M. L., SELESON, P., PLIMPTON, S. J., SILLING, S. A., and LEHOUCQ, R. B. *Peridynamics with LAMMPS: a User Guide v*0.3 *Beta*, Sandia Report, 3532 (2011)

[11] MITCHELL, J. A. *A Non-local, Ordinary-state-based Viscoelasticity Model for Peridynamics*, Sandia National Lab Report, 8064 (2011)

[12] KIM, M., WINOVICH, N., LIN, G., and JEONG, W. Peri-net: analysis of crack patterns using deep neural networks. *Journal of Peridynamics and Nonlocal Modeling*, **1**(2), 131–142 (2019)

[13] GARNELO, M., SCHWARZ, J., ROSENBAUM, D., VIOLA, F., REZENDE, D. J., ESLAMI, S. M., and TEH, Y. W. Neural processes. *arXiv Preprint*, arXiv: 1807.01622 (2018) https://doi.org/10.48550/arXiv.1807.01622

[14] PARKS, M. L., LEHOUCQ, R. B., PLIMPTON, S. J., and SILLING, S. A. Implementing peridynamics within a molecular dynamics code. *Computer Physics Communications*, **179**(11), 777–783 (2008)

[15] RASMUSSEN, C. E. Gaussian processes in machine learning. *Advanced Lectures on Machine Learning*, Springer, Berlin/Heidelberg, 63–71 (2004)

[16] PLIMPTON, S. J. Pizza.py (2022) http://www.cs.sandia.gov/sjplimp/pizza.html

[17] AHRENS, J., GEVECI, B., and LAW, C. ParaView: an end-user tool for large data visualization. *Visualization Handbook*, Elsevier, Amsterdam (2005)

[18] AYACHIT, U. *The ParaView Guide*: *a Parallel Visualization Application*, Kitware, New York (2015)

[19] VEDALDI, A. and LENC, K. MatConvNet: convolutional neural networks for MATLAB. *Proceedings of the* 23*rd ACM International Conference on Multimedia*, Association for Computing Machinery, New York, 689–692 (2014)

[20] PASZKE, A., GROSS, S., CHINTALA, S., CHANAN, G., YANG, E., DEVITO, Z., LIN, Z., DESMAISON, A., ANTIGA, L., and LERER, A. Automatic differentiation in PyTorch. *NIPS* 2017 *Autodiff Workshop* (2017) https://openreview.net/forum?id=BJJsrmfCZ

[21] LECUN, Y., BOTTOU, L., BENGIO, Y., and HAFFNER, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, **86**(11), 2278–2324 (1998)

[22] GOODFELLOW, I., BENGIO, Y., COURVILLE, A., and BENGIO, Y. *Deep Learning*, Vol. 1, MIT Press, Cambridge (2016)

[23] GERON, A. *Hands-on Machine Learning with Scikit-Learn and TensorFlow*: *Concepts*, *Tools*, *and Techniques to Build Intelligent Systems*, 'O'Reilly Media, Sebastopol (2017)

[24] LECUN, Y., BENGIO, Y., and HINTON, G. Deep learning. *nature*, **521**(7553), 436–444 (2015)

[25] KRIZHEVSKY, A., SUTSKEVER, I., and HINTON, G. E. ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, **60**, 84–90 (2017)

[26] NASRABADI, N. M. Pattern recognition and machine learning. *Journal of Electronic Imaging*, **16**(4), 049901 (2007)