DATA-DRIVEN MULTISCALE PREDICTION OF MATERIAL PROPERTIES USING MACHINE LEARNING ALGORITHMS

by

Moonseop Kim

A Dissertation

Submitted to the Faculty of Purdue University

In Partial Fulfillment of the Requirements for the degree of

Doctor of Philosophy

School of Mechanical Engineering

West Lafayette, Indiana

December 2019

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF COMMITTEE APPROVAL

Dr. Guang Lin, Chair
Department of Mechanical Engineering
Dr. Liang Pan
Department of Mechanical Engineering
Dr. Xiulin Ruan
Department of Mechanical Engineering
Dr. Peilin Liao
Department of Materials Engineering

Approved by:
Dr. Jay Gore
Head of the Graduate Program

To my wife and family

ACKNOWLEDGEMENTS

First of all, I would like to thank Professor Guang Lin, who helped me to do my Ph.D. He not only helped me a lot in my research but also gave me a chance to do my research well during my doctorate. I want to repeat thanks. Also, I would like to say thanks to my committee members, Professor Liang Pan, Peilin Liao, Kejie Zhao, and Xiulin Ruan. I want to thank Professor Byeong-Chan Lee and Professor Junemo Koo, who helped us to come to the US.

I specially thanks to my Soojung Kim, my dearest wife, who was the greatest strength during my doctoral course. She gave me the best support for my Ph.D., and always gave me a lot of encouragement and praise, helping me to do my Ph.D. Without her, I would not have finished my Ph.D. She is also a Ph.D. student in Special Education now, and I hope she will finish well during the rest of the year, and I will give her the best support. I want to say thank you and love you again and please grow strong my little baby (Trong Kim) and do not make mom hard.

I want to say thank you once again to my parents (Youngkun Kim and Hyesoon Min), who gave me unconditional support until now. My parents gave me a lot of support and encouragement to obtain this position. Thanks again and now I think it is time to repay an obligation. I want to say thank you and love you.

I would also like to express my gratitude to my father-in-law (Sookyun Kim) and my mother-in-law (Jeongsun Ko), who gave me lots of help and encouragement during my doctorate. I want to express my appreciation to them for giving me my most beloved wife and for giving me unconditional support while I am in the US, and thanks to my brother-in-law (Bokyung Kim), his wife (Joohee Son) and his son (Won Kim).

I would also like to thank all my relatives, who supported my doctoral program. I want to express my gratitude to my school mate, Heetaek Yoon, Wonchan Sung, Jaesang Hyun, and Hye-ran Moon. I want to express my gratitude to my senior and junior colleagues Nicholas Kim, Hyunjun Shin, Janghyun Kim, Bongjoong Kim, and Woohyun Park. I want to say thank you all.

Finally, I would like to express my gratitude to my best friends Yuh-Keun Yoon, Jae Kwang Jeong, Jaeeun Hwang, Sanghun Shin, Junho Kang, Seungwon Lee, Jaeyoon Kim, Kwangju Baek, Jaeho Bae.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Author: Kim, Moonseop. PhD
Institution: Purdue University
Degree Received: December 2019
Title: Enhancement of Mechanical Properties and Analysis of Damage Using Molecular Dynamics Simulations with Machine Learning
Committee Chair: Guang Lin

The objective of this study is that combination of molecular dynamics (MD) simulations and machine learning to complement each other. In this study, four steps are conducted.

First is based on the empirical potentials development in silicon nanowires for theory parts of molecular dynamics. Many-body empirical potentials have been developed for the last three decades, and with the advance of supercomputers, these potentials are expected to be even more useful for the next three decades. Atomistic calculations using empirical potentials can be particularly useful in understanding the structural aspects of Si or Si-H systems, however, existing empirical potentials have many errors of parameters. We propose a novel technique to understand and construct interatomic potentials with an emphasis on parameter fitting, in which the relationship between material properties and potential parameters is explained. The input database has been obtained from density functional theory (DFT) calculations with the Vienna ab initio simulation package (VASP) using the projector augmented-wave method within the generalized gradient approximation. The DFT data are used in the fitting process to guarantee the compatibility within the context of multiscale modeling.

Second, application part of MD simulations, enhancement of mechanical properties was focused in this research by using MEAM potentials. For instance, Young's modulus, ultimate tensile strength, true strain, true stress and stress-strain relationship were calculated for nanosized Cu-precipitates using quenching & partitioning (Q&P) processing and nanosized Fe_3C strengthened ultrafine-grained (UFG) ferritic steel. In the stress-strain relationship, the structure of simulation is defined using the constant total number of particles, constant-energy, constant-volume ensemble (NVE) is pulled in the y-direction, or perpendicular to the boundary interface, to increase strain. The strain in increased for a specified number of times in a loop and the stress is calculated at each point before the simulation loops.

Third, based on the MD simulations, machine learning and the peridynamics are applied to prediction of disk damage patterns. The peridynamics is the nonlocal extension of classical continuum mechanics and same as MD model. Especially, FEM is based on the partial differential equations, however, partial derivatives do not exist on crack and damage surfaces. To complement this problem, the peridynamics was used which is based on the integral equations and overcome deficiencies in the modeling of deformation discontinuities. In this study, the forward problem (i), if we have images of damage and crack, crack patterns are predicted by using trained data compared to true solutions which are hit by changing the x and y hitting coordinates on the disk. The inverse problem (ii), if we have images of damage and crack, the corresponding hitting location, indenter velocity and indenter size are predicted by using trained data. Furthermore, we did the regression analysis for the images of the crack patterns with Neural processes to predict the crack patterns. In the regression problem, by representing the results of the variance according to the epochs, it can be confirmed that the result of the variance is decreased by increasing the epoch through the neural processes. Therefore, the result of the training gradually improves, and the ranges of the variance are expressed as 0 to 0.035. The most critical point of this study is that the neural processes makes an accurate prediction even if the information of the training data is missing or not enough. The results show that if the context points are set to 10, 100, 300, and 784, the training information is deliberately omitted such as context points of 10, 100 and 300, and the predictions are different when context points are significantly lower. However, when comparing the results of context points 100 and 784, the predicted results appear to be very similar to each other because of the Gaussian processes in the neural processes. Therefore, if the training data is trained through the neural processes, the missing information of training data can be supplemented to predict the results.

Finally, we predicted the data by applying various data using deep learning as well as MD simulation data. This study applied the deep learning to Cryo-EM images and Line Trip (LT) data with power systems. In this study, deep learning method was applied to reduce the effort of selection of high-quality particles. This study proposes a learning frame structure using deep learning and aims at freeing passively selecting high quality particles as the ultimate goal. For predicting the line trip data and bad data detection, we choose to analyze the frequency signal because suddenly the frequency changes in the power system due to events such as generator trip, line trip or load shedding in large power systems.

CHAPTER 1. INTRODUCTION

In the last three decades, empirical potentials have been advanced. With the advance of supercomputers, these potentials are anticipated to be more useful for the next three decades [1]. Atomistic calculations by empirical potentials can be utilize in understanding the structural aspects of Si or Si-H systems that are found in many important areas such as the surface of nano patterning Si [2, 3], nano-electro-mechanical systems (NEMS) [4], superconductivity of silane [5], optical modulators [6], and applications of α-Si:H materials [7]. In the past, empirical potentials for Si [8-11] and for Si-H [12-14] have been developed constantly, but none of them have been resolved the bulk elastic properties of Si. In Ref.12, they show the different hydrogen-induced reconstructions of the silicon surface then distance between hydrogen and hydrogen is 1.64 Å and bond angle H-Si-H is 106° using existing empirical potential, however, when H-H distance and bond angle are compared to results of first principle calculations, H-H distance and bond angle are 2.1638 Å and 104.805 ° respectively. In this situation, bond angle is distinguished from 1.195° which means that difference of bond angle can be ignored, however, the biggest problem is that H-H distance is distinguished from 0.5238 Å. In this respect, if existing empirical potential are used for Si nanowires, computation speed is fast but not accuracy compared to results of first principle calculations, the other problem is that it includes errors autonomously. It is essential to fix errors preventing even much bigger errors. In this paper, we propose a novel technique to understand and construct empirical potentials with an emphasis on parameter fitting, in which the relationship between material properties and potential parameters is explained. The input database has been obtained from density functional theory (DFT) calculations with the Vienna ab initio simulation package (VASP) [15]. For application part of MD simulations, based on empirical potential, enhancement of mechanical properties was focused in this research by using MEAM potentials. For instance, Young's modulus, ultimate tensile strength, true strain, true stress and stress-strain relationship were calculated for nanosized Cu-precipitates using quenching & partitioning (Q&P) processing and nanosized Fe_3C strengthened ultrafine-grained (UFG) ferritic steel. In an effort to reduce vehicle weight but still satisfying the more stringent safety regulations, the automotive industry world-wide has been adopting various types of advanced high strength steels (AHSS) in the past decades. It is well established that a good combination of strength and ductility can be achieved by following factors: solid solution strengthening and precipitation hardening, transformation strengthening, grain boundary strengthening, and back stress hardening resulted from gradient structure [16-19]. Transformation-induced-plasticity (TRIP) steel has been used in the automotive body construction due to their outstanding combination of strength and ductility. The strengthening mechanism for TRIP-assisted steels is based on the deformation-induced transformation of the metastable austenite, a face-centered cubic (fcc) γ-phase characterized by a high work hardening capacity to achieve very high tensile strengths with exceptional ductility [18], to martensite characterized by a hexagonal close-packed (hcp) ε-phase and/or body-centered cubic (bcc) α′-phase, during the deformation process [19, 20]. The TRIP effect can significantly improve the formability and energy absorption of this class of materials [21]. The quenching and partitioning (Q&P) processing has been proved to be an effective method for achieving a good combination of strength and ductility in steels [22, 23]. In this study, the morphology and volume fraction of Cu precipitates and all phases in the steels after different O&P processing have been characterized using numerical methods based on molecular dynamical simulations. Next, we mainly focused on characterizing the microstructures and mechanical properties of low alloy medium-carbon steel with a duplex microstructure composed of nanoscale spheroidized Fe3C in an ultrafine-grained (UFG) ferritic steel. Molecular dynamic (MD) simulation based on Modified embedded-atom method (MEAM) was employed to explain the strengthening effect of nanosized Fe3C precipitates in Ferrite-Fe3C system. This study has provided useful insights into the mechanisms contributing to work hardening of the UFG steels strengthened using nanoscale precipitates. It is expectable that the present results may be beneficial for the design of new steels and the modeling of the corresponding deformation 15ehavior. Finally, based on the MD simulations, machine learning and peridynamics are applied to prediction of disk damage patterns. Many products in everyday life are damaged by external shocks or user's mistakes, and cracks are formed. In the Department Of Energy (DOE), it is often the case that blades of thermoelectric power plant and wind power plant are damaged by rotation, temperature effects, and external wind influences. In the reactor of nuclear power plant, cracks are occurred due to heat generated during nuclear fission. This can lead to not only high maintenance costs but also national disaster. So far, research on cracking has been going on in various perspectives for several decades. Representatively, various types of structures and material behaviors have been studied through the finite element analysis (FEM). However, there are inherent errors in the study through the FEM. In the FEM, only the approximate solution can be obtained. Also, the results of analysis tend to depend on the mesh, and the mistakes by the researchers can be fatal to derive the results. Finally, the FEM is based on partial differential equations, but the partial derivatives are not present on the surface of the crack, so it is not enough to get accurate results. Therefore, in order to compensate for these drawbacks, we conducted a crack pattern study using the peridynamics. The peridynamics theory of solid mechanics was first introduced by S. Silling [24-27]. This theory is nonlocal extension of classical continuum mechanics and It is an alternative to continuum mechanics for more accurate crack studies. This model is based on integral equations and the integral equations of the peridynamics can be applied directly to cracks. The reason for this is that it does not require partial derivatives. Therefore, the peridynamics is very suitable for the study of surface discontinuity such as cracks, and various multiscale modeling is possible due to SI units can be used. The peridynamics can be of particular use in understanding the damage in membranes and nanofiber [28], composites and brittle materials [29], brazed single-lap joints [30], fuel pellet [31] and biomembranes [32]. The peridynamics can be applied not only to damage studies, but also to technologically important areas such as prediction of viscoelastic materials [33], piezo-resistive response of carbon nanotube nanocomposites [34], phase transformation in zirconium dioxide [35], shock and vibration [36], and indentation of thin copper film [37]. Though the peridynamics framework has proven to be a powerful and widely applicable tool, the computational demand can become burdensome very quickly as the scales of the simulations are increased. In order to decrease the computation time required to run these simulations while maintaining a high level of accuracy, we propose a machine learning approach which utilizes recent design and hardware advances that have greatly improved the performance of artificial neural networks (ANNs) [38-39]. Recently, convolutional neural networks (CNNs) have been shown to provide excellent light-weight alternative to traditional fully-connected networks [40] and are particularly well-suited for applications on spatially structured data [41], which is characteristic of the physical systems considered in the peridynamics. These networks have the advantage of being trainable in advance, learning from a dataset generated by a highly accurate model such as the peridynamics, and encoding an approximation of the model into a concise, neural network representation [42]; this approximate model can then be used after the training procedure to produce near instantaneous results. By combining the accuracy of the peridynamics model with the computational speed of modern neural networks, we show that complex MD simulations can be accurately approximated using just a fraction of the computation time required by traditional approaches. Furthermore, we applied the neural processes (NPs) [80] which combine the advantages of the neural networks with the advantages of a Gaussian process with the data obtained from the peridynamics theory. Therefore, we can learn how to apply advanced information to the data, improve the efficiency of computation during the training and evaluation resulting in fast and more accurate results. Finally, in this study, the CNNs and the neural processes are applied based on the data obtained from the peridynamics theory. As a result, we are trying to develop the peridynamics theory even more precisely with the case study for the characteristic of materials, and by applying the deep machine learning, we can speed up the calculation speed using the trained data and finally reduce the computational cost as a result.

Finally, this study applied deep learning using various data such as Cryo-EM images and Line Trip (LT) data with power systems. Recent developments in Cryo-EM technology have had a great impact on structural biology and protein molecular complexes. However, high-resolution Cryo-EM studies of molecular complexes require the choice of a large number of high-quality particles. The particle selection stage in this process is very labor intensive. This is because it is manually selected by the human hand to obtain high quality particles. Also, Particle selection is also very subjective because it is done passively, and the results can be very different because particle selection is done by humans. Therefore, in this study, deep learning method was applied to reduce the effort of selection of high-quality particles. In recent years, deep learning technology has been applied to researches such as computer vision and natural language processing through the availability of large – scale learning data, the development of powerful computing platforms and learning algorithms. Therefore, this study proposes a learning frame structure using deep learning and aims at freeing passively selecting high quality particles as the ultimate goal. In the problem of the power system, Continuous or simultaneous failures / events in power systems are a common problem that can lead to frequent section blackouts. In recent decades, many researches have been reported on perturbation or event detection and perception. Various types of signals were used for analysis purposes, including frequency, power and voltage, and phasor angles. In this paper, we choose to analyze the frequency signal because suddenly the frequency changes in the power system due to events such as generator trip, line trip or load shedding in large power systems.

CHAPTER 2. EMPIRICAL POTENTIAL DEVELOPMENT FOR SiNWs

2.1 Structure of SI:H Nanowires

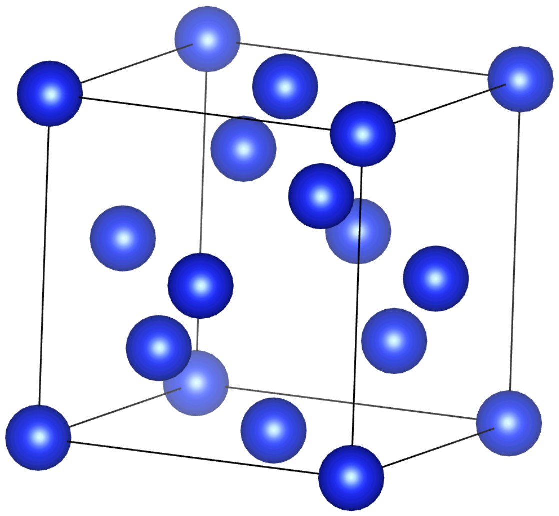


Fig. 2.1. Diamond cubic structure.

Before describing the structure in which hydrogen is passivated to the silicon nanowires in this study, we first examine the structure of silicon. Fig. 2.1 is a diagram illustrating the crystal structure of silicon as a diamond cubic crystal structure. The diamond cubic crystal structure is a structure in which four atoms are settled in the shape of a face centered cubic crystal structure. Let's take the example of cutting this structure to the length and size that we want and use it for experiments. If we cut the silicon to the desired shape or length, the surface will have dangling bond. This would lead to corrosion of the silicon nanowires if the dangling bond reacts with oxygen. Therefore, this study stabilizes the surface of silicon nanowires by passivating hydrogen to the silicon nanowires. In other words, the surface structure of the silicon nanowires will passivate all the dangling bonds of the wire surface to hydrogen atoms to prevent corrosion of the surface of the nanowire. By passivating the hydrogen, it became a role to eliminate the instability of the surface of the wire.

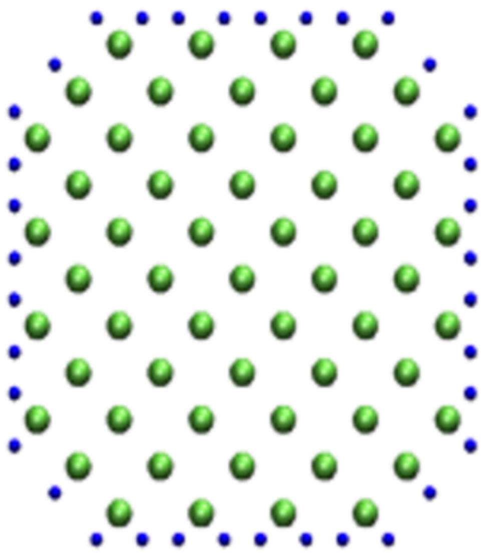


Fig. 2.2. Cross section of silicon nanowires passivated hydrogen <001>.

In this study, Si nanowires passivated hydrogen model is chosen. Fig. 2.2 [43] has expressed a cross section of silicon nanowires. Green dots and blue dots are expressed silicon atoms and hydrogen atoms respectively. Fig. 2.2 is represented by structure of Si nanowires and cross section of Si nanowires is Wulff structure is selected by minimizing surface energy. It can be more stabilized by passivating hydrogen to surface of Si nanowires. In mechanical property part, Young's modulus and equilibrium elongation will be calculated after H-H parameters fitting by increasing the size of cross section compared to results of Young's modulus and equilibrium elongation by using existing empirical potentials and first principles calculations.

2.2 Govern Equation of Tersoff Empirical Potentials

The atomistic computer simulations based on empirical potential is fast for speed of calculation. In this system, the number of atoms is not limited compared to first principles, however, the accuracy of calculation is not adequate, therefore, reliability of empirical potential presented so far are needed to verify. There are various empirical potentials depending on the type of material, for instance, Nickel (Ni) and Titanium (Ti) are calculated through EAM (Embedded Atom Method) [44] and Silicon (Si) is calculated through Tersoff empirical potential [9] and Stillinger-Weber empirical potential [8]. In this study, silicon nanowires passivated hydrogen for the model and Tersoff empirical potential are used for the purpose of verifying accuracy of existing Tersoff empirical potential. Tersoff empirical potential is based on the concept of bond order, force of bonds between atoms is not consistent and depends on local environment. Tersoff empirical potential expands concept and form of existing variously. The total energy function is given as [12]

$$V = 0.5 * \sum_{\substack{i,j \\ i \neq j}} f_R(r) + b_{ij} f_A(r) \tag{1}$$

$$f_R(r) = Aexp(-\lambda_1 r_{ij}) \tag{2}$$

$$f_A(r) = -Bexp(-\lambda_2 r_{ij}) \tag{3}$$

$$b_{ij} = (1 + \zeta_{ij}^{\eta})^{-\delta} \tag{4}$$

V is total energy function, $f_R(r)$ and $f_A(r)$ are repulsive energy and attractive energy respectively. These functions are defined as function of distance between i and j atoms, r is interatomic distance and b_{ij} is bond order. In this study, A, B, λ_1 and λ_2 are decided as H-H parameters fitting.

$$\zeta_{ij} = \sum_{k \neq i,j} f_c(r_{ik})\left[c + d\{H(N) - cos\theta_{ijk}\}^2\right] \times \exp\left[\alpha\{(r_{ij} - R_{ij}^e) - (r_{ik} - R_{ik}^e)\}^{\beta}\right] \tag{5}$$

ζ_{ij} is function of effective coordination number, H(N) is function of bond number, $cos\theta_{ijk}$ is bond angle, r_{ij} and r_{ik} are distance between i and j atoms and between i and k respectively and R_{ij}^e and R_{ik}^e are equilibrium distance between i and j atoms and between i and k respectively. Lastly, in this study, $\alpha,\ \beta,\ \eta,\ \delta$ and c are decided as H_2-H_2 parameters fitting.

$$f_c(n) = \begin{cases} 1, & r_{ij} < R - D \\ \frac{1}{2} - \frac{9}{16}\sin\left(\pi \frac{r_{ij} - R}{2D}\right) - \frac{1}{16}\sin\left(3\pi \frac{r_{ij} - R}{2D}\right), & R - D < r_{ij} < R + D \\ 0, & r_{ij} > R + D \end{cases} \tag{6}$$

$f_c(n)$ is cutoff function which is judging whether there is coherence or not between atom and neighbor atom. R is interatomic distance, R and D are a constant to determine range of inter-atomic influence. According to R and D value, it is possible to judge obtaining numerical value. If interatomic distance is less than R-D, the influence is 1, if interatomic distance is larger than R-D, the influence is 0. Finally, if interatomic distance is between R-D and R+D, it is influenced by Eq. (5).

2.3 Multi-fidelity Gaussian Process Regression for Prediction of High Fidelity to Increase the Number of Modes in Low Fidelity

This is the standard setup for multi-fidelity modelling with Gaussian processes. The steps on multi-fidelity are given as

$$u_1(x) \sim \mathcal{GP}(0, k_1(x, x'; \theta_1)) \tag{7}$$

$$u_2(x) \sim \mathcal{GP}(0, k_2(x, x'; \theta_2)) \tag{8}$$

$u_1(x)$ and $u_2(x)$ are independent and in Gaussian process regression, it is assumed that the mean of $\mathcal{GP}$ is zero and $k(x, x'; \theta)$ is the covariance matrix between all possible pairs (x, x') in set of vector of hyper-parameters θ. [44] The basic idea is that we begin with two independent GP $u_1(x)$ and $u_2(x)$; then we define low fidelity and high fidelity models: [45-50]

$$f_L(x) = u_1(x), \; f_H(x) = \rho u_1(x) + u_2(x) \tag{9}$$

This "relationship" the low and high-fidelity models since both include the GP $u_1(x)$. In particular setting, k_1=cov[u_1, u_1] and k_2=cov[u_2, u_2] we have:

$$K_{LL} = \text{cov}[f_L, f_L] = \text{cov}[u_1, u_1] = k_1 \tag{10}$$

$$K_{LH} = \text{cov}[f_H, f_L] = \text{cov}[\rho u_1 + u_2, u_1] = \rho\text{cov}[u_1, u_1] + \text{cov}[u_2, u_1] = \rho k_1 \tag{11}$$

$$K_{HH} = \text{cov}[f_H, f_H] = \text{cov}[\rho u_1 + u_2, \rho u_1 + u_2] = \rho^2\text{cov}[u_1, u_1] + \rho\text{cov}[u_2, u_1] + \rho\text{cov}[u_1, u_2] + \rho\text{cov}[u_2, u_2] = \rho^2 k_1 + k_2 \tag{12}$$

cov[u_1, u_2] = 0 and cov[u_2, u_1] = 0 by independence and to sum up K_{LL}, K_{LH}, K_{HH}:

$$\begin{aligned} K_{LL}(x,x') &= k_1(x,x';\theta_1) \\ K_{LH}(x,x') &= \rho k_1(x,x';\theta_1) \\ K_{HH}(x,x') &= \rho^2 k_1(x,x';\theta_1) + k_2(x,x';\theta_2) \end{aligned} \tag{13}$$

This gives us a complete model which incorporates both the low and high-fidelity. In particular, we model the column vector $[f_L(x); f_H(x)]$ using a zero-mean prior and the covariance matrix defined block-wise by: $[\ K_{LL}, K_{LH}; K_{HL}, K_{HH}]$. Since the mean and covariance are known, the whole Gaussian process model is specified and training will begin using the standard procedure.

$$\begin{bmatrix} f_L(x) \\ f_H(x) \end{bmatrix} \sim \mathcal{GP}\left(0, \begin{bmatrix} K_{LL}(x,x') & K_{LH}(x,x') \\ K_{HL}(x,x') & K_{HH}(x,x') \end{bmatrix}\right) \tag{14}$$

In training, based on $\{x_L, y_L\}$, $\{x_L, y_L\}$, N_L>> N_H,

$$\begin{bmatrix} y_L(x) \\ y_H(x) \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} K_{LL}(x_L,x_L) + \sigma_L^2 I & K_{LH}(x_L,x_H) \\ K_{HL}(x_H,x_L) & K_{LL}(x_H,x_H) + \sigma_H^2 I \end{bmatrix}\right) \tag{15}$$

This matrix adds the noise by including the terms on the diagonal of the covariance matrix. This just adds uncertainty (or noise) into the way the observations are included into the model.

$$\text{NLML}(\theta_1, \theta_2, \rho) = \frac{1}{2} y^{\mathrm{T}} \mathcal{K}^{-1} \mathrm{y} + \frac{1}{2} \log|\mathcal{K}| + \frac{N_L + N_H}{2} \log(2\pi) \tag{16}$$

$\mathcal{K}$ is the covariance matrix and the Negative Log Marginal Likelihood (NLML) is used as the "cost function" which should be minimized to get the best-fit model by using hyper-parameters θ. In prediction, if we consider a Gaussian likelihood and the posterior distribution is easy to apply and can be used to involve predictive deduction for a new output f_H, given a new input x* as

$$f_H(x^*|y \sim \mathcal{N}([K_{HL}(x^*, x_L)\ K_{HH}(x^*, x_H)]\mathcal{K}^{-1}\mathrm{y}, K_{HH}(x^*, x^*) - K(x^*, x)\mathcal{K}^{-1}\ K(x, x^*)\,) \tag{17}$$

In numerical results, to reduce the computational cost for expensive calculations (DFT), Multi-fidelity Gaussian process regression for prediction [45-50] is used. H-H binding energy and H_2-H_2 interaction energy with empirical potential are applied to low-fidelity model and results of

H-H binding energy and H_2-H_2 interaction energy with DFT are implied to high-fidelity model for few samples.

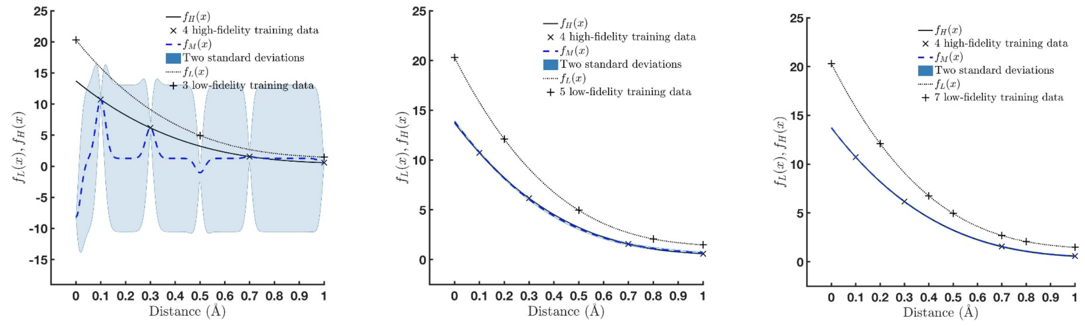


Fig. 2.3. The interaction energy of $f_H(x)$ is obtained with high-fidelity samples (N_H=4) and increase the number of low-fidelity modes. If we increase N_L, the accuracy is increased (the standard deviation of the high-fidelity decreases).

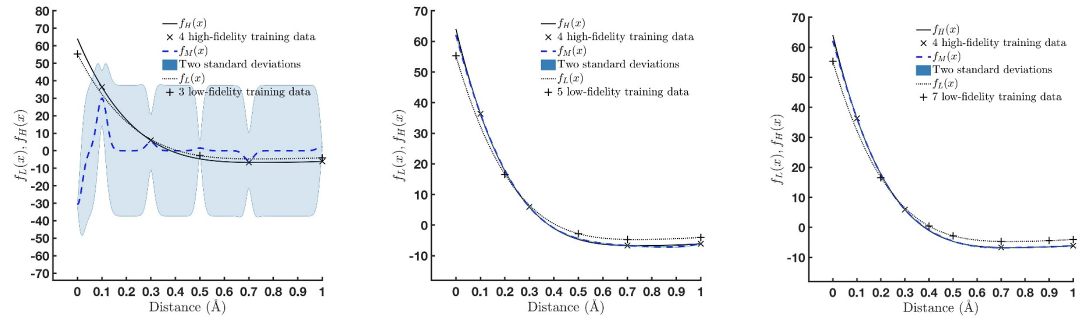


Fig. 2.4. The binding energy of $f_H(x)$ is obtained with high-fidelity samples (N_H=4) and increase the number of low-fidelity modes. If we increase N_L, the accuracy is increased (the standard deviation of the high-fidelity decreases).

In Fig. 2.3 and Fig. 2.4, high-fidelity samples (N_H=4) are fixed and we compare the standard deviation of the high-fidelity to increase the number of low-fidelity modes. As we can see, the standard deviation of the high-fidelity is decreased when the number of low-fidelity modes are increased.

2.4 H-H Fitting Parameters

2.4.1 H-H Parameters Fitting Method (Nelder-Mead Method)

In this study, various fitting methods were used to minimize the difference of binding energy and interaction energy by using first principles and empirical potential. Let’s look at the Nelder – Mead method [51-52] that minimizes energy differences in various fitting methods. This method is proposed by John Nelder and Roger Mead as a way to find the local minimum of a function containing multiple variables by applying an initial value to the function. This method is generally used to solve non-linear optimization problems. The Nelder-Mead method is as follows. First, define the function you want to minimize f (x, y) and specify three vertices of the initial triangle to start this method.

$$V_k = (x_k, y_k)\,, \qquad k = 1,2,3 \tag{18}$$

Put these three points into the function and rearrange them in the order of smaller results.

$$B = (x_1, y_1), G = (x_2, y_2), W = (x_3, y_3) \tag{19}$$

B, G and W define the best vertex, good vertex, the worst vertex respectively. Next, the center point M is set between B and G. M is defined as follows.

$$M = \frac{B+G}{2} = \left(\frac{x_1+x_2}{2}, \frac{y_1+y_2}{2}\right) \tag{20}$$

The function decreases gradually from the point W to the point B and from the point W to the point G along the sides of the triangle in the direction of lower energy. Therefore, since the function f (x, y) tends to go in the direction of decreasing the energy, the worst point W reflects to a symmetrical R point centering around the line connecting the B and

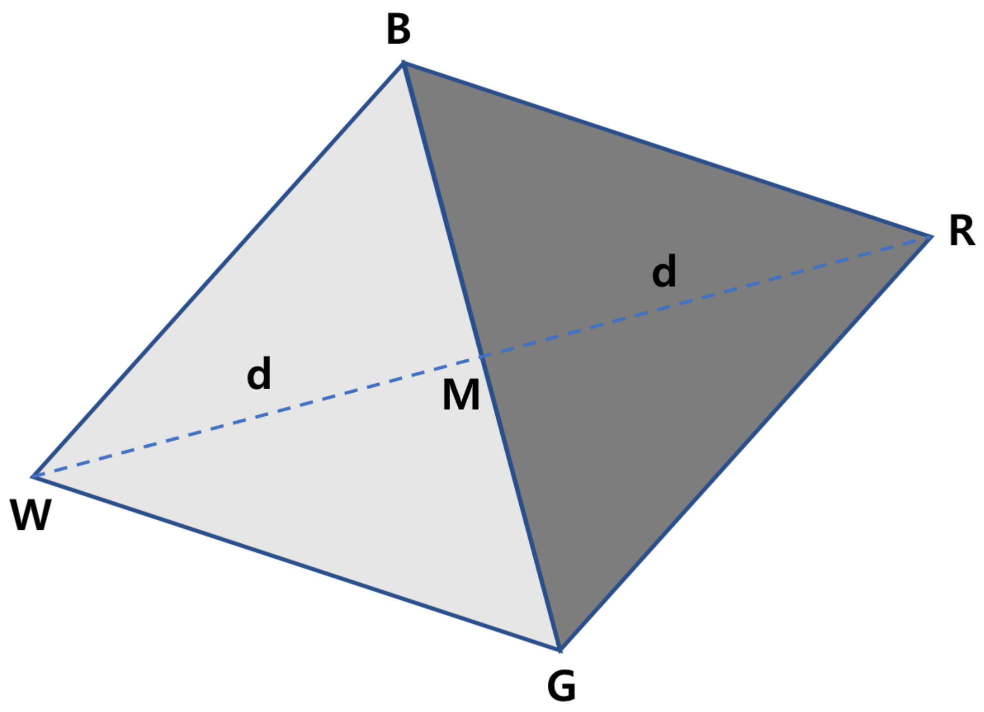


Fig. 2.5. Reflection using the point R.

G points. In order to determine the point R, the midpoint M between B and G is found first. The distance between point W and point M is d. The point moved from the M point to the opposite point by d is the R point and R is defined as follows.

$$R = 2M - W \tag{21}$$

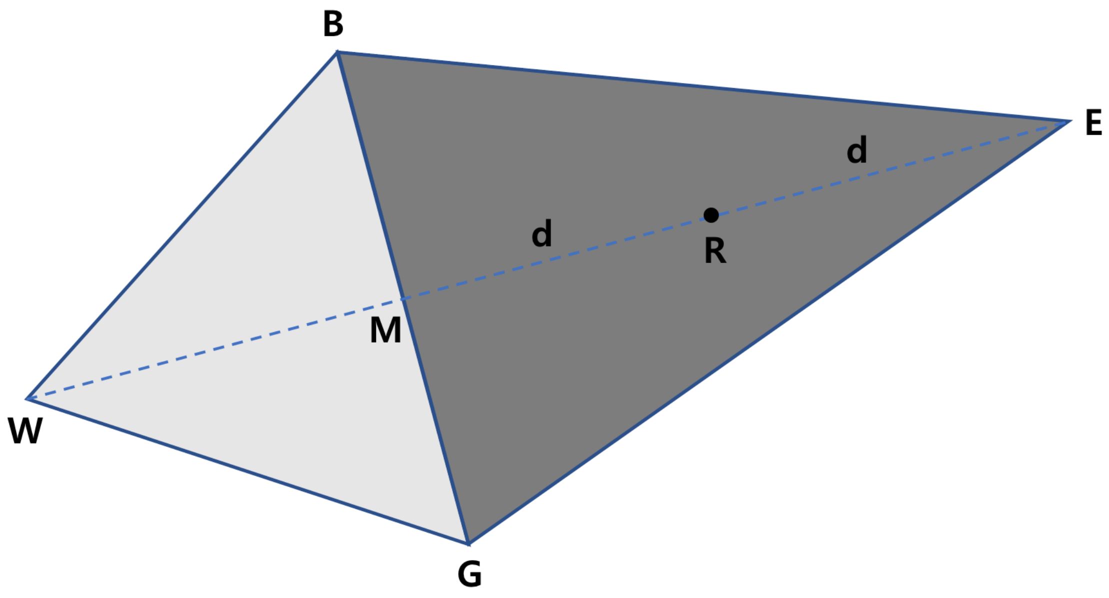


Fig. 2.6. Expansion using the point E.

If the function value of the R is smaller than the function value of the W, the Nelder-Mead method will induce toward the minimum. The minimum value can be further from point R. we extend the line segment through M and R to the point E. The point E is found by moving an additional distance d along the line joining M and R. If the function value at E is less than the function value at R, then we have found a better vertex than R. The vector formula for E is

$$E = 2R - M \tag{22}$$

So far, we have studied the Nelder-Mead method. The following Table 2.1 is an explanation of the algorithm of the Nelder-Mead method based on the above-mentioned description.

Table 2.1. Algorithm for the Nelder-Mead method.

```
IF f(R)<f(G), THEN perform Case(i) {either reflect or extend}
ELSE perform Case(ii) {either contract or shrink}
```

```
BEGIN {Case (i)}
IF f(B)< f(R) THEN
    replace W with R
  ELSE
  Compute E and f(E)
  IF f(E)<f(B) THEN
    replace W with E
  ELSE
    replace W with R
  ENDIF
ENDIF
END {Case(i)}
```

```
BEGIN {Case (ii)}
IF f(R)<f(W) THEN
    replace W with R
Compute C=(W+M)/2
or C=(M+R)/2 and f(C)
  IF f(C)<f(W) THEN
      replace W with C
  ELSE
      Compute S and f(S)
    replace W with S
    replace G with M
  ENDIF
END{Case(ii)}
```

If the value of f(R) is smaller than the value of f(G) as a result of deriving the function value using the above-mentioned contents, the case (i) can be calculated. If f(B) is less than f(R), replace W with R and if not, calculate the values of E and f(E). Also replace W with E if f(B) is less than f(R) and f(E) is less than f(B), and if not, replace W with R. If the algorithm is continuously repeated in this way, the final results can be obtained by obtaining a result that converges at all of B, G, and W points. As I mentioned part 2.1, the structure of Si nanowires is passivated by hydrogen to prevent oxidation. This model can divide three parts of the Si nanowires for parameters fitting. First, Si-Si parameters fitting are needed for internal structure of Si nanowires, Second, H-H parameters fitting are needed for surface computation, Lastly, Si-H parameters for interface calculation are needed.

Table 2.2. H-H parameters for molecular hydrogen compared three optimization methods which are Nelder-Mead Simplex Method (N-M) [52], Broyden-Fletcher-Goldfarb-Shanno Quasi-Newton Method (BFGS) [53–56], Trust-Region Method (T-R) [57, 58] and RMSE (Root Mean Square Error) is defined as $\mathrm{RSME} = \sqrt{\frac{1}{N_a}\sum_{a=1}^{N_a}(V_a - E_a)^2}$, V-E is difference of binging Energy and Interaction Energy are calculated to DFT (V) Energy and Empirical potential (E) Energy respectively.

	N-M	BFGS	T-R	Existing
A	87.5482	87.5471	80.0752	80.07
B	18.2497	18.2498	31.3714	31.38
λ_1	5.2898	5.2898	3.9932	4.2075
λ_2	1.0290	1.0290	1.4134	1.7956
R^e	0.75	0.75	0.75	0.74
R	3.5	3.5	3.5	1.4
D	0.5	0.5	0.5	0.3
h	0	0	0	0
d	0	0	0	0
α	3	3	3	3
β	1	1	1	1
c	1.3328	1.3322	3.9550	4
δ	0.6923	0.6926	0.3576	0.A80469
η	1.2866	1.2865	0.8464	1
RMSE (Binding Energy)	0.0014	0.0014	0.1341	
RMSE (Interaction Energy)	0.0012	0.0013	0.0521	

In this study, the H-H parameters fitting is focused by using H-H binding energy and H_2-H_2 interaction energy for surface computation. In surface of Si nanowires, there are two forces for hydrogen relations that are attractive and repulsive forces. This H-H parameter fitting is complicated so we have to design systematically. When we fit H-H parameters fitting by using H-H binding energy, we only use Eq. (1) – Eq. (3) except for Eq. (4, 5). This is because H-H binding energy is calculated by two hydrogen atoms which can be neglected ζ_{ij} term that needs more than three atoms, however, H_2-H_2 parameters fitting by using interaction energy are applied to more than three hydrogen atoms so it should be applied ζ_{ij} term. In Table. 2.2, three methods of optimization were used for H-H parameters fitting. Lastly, we can find the error of each method between DFT results and fitting line through root mean square error that represents on Table. 2.2. Nelder-Mead simplex method is the best method for error compare to other methods. The parameters using Nelder-Mead simplex method were chosen are listed on Table. 2.2.

2.4.2 H-H Parameters Fitting by Using Binding Energy

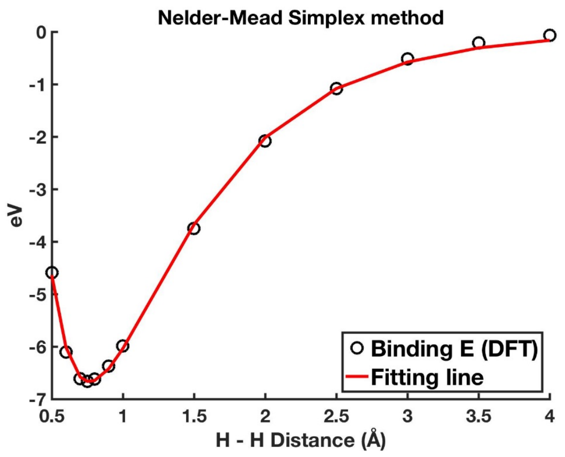


Fig. 2.7. H-H parameters fitting by using Nelder-Mead Simplex method.

In the study, we compute the binding energy using DFT for reference results that are adjusted to the results of empirical potential's objective function and parameters can be obtained after fitting between hydrogen and hydrogen. Fig. 2.7 shows that black dots explain DFT results for H-H binding energy and red line express fitting line. A, B, λ_1, λ_2 and R^e values are listed on Table 2.2.

2.4.3 H_2-H_2 Parameters Fitting by Using Interaction Energy

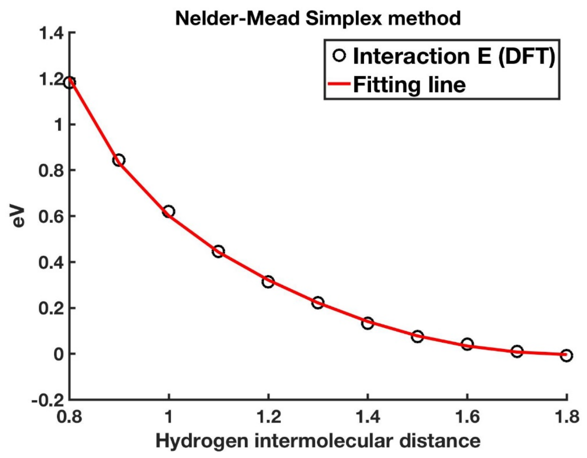


Fig. 2.8. H_2-H_2 Parameters Fitting by using Nelder-Mead Simplex method.

In this section, we compute interaction energy using DFT for reference results that are adjusted to the results of empirical potential's objective function and we obtained parameters after fitting hydrogen intermolecular. We use Nelder-Mead Simplex methods of optimization for H-H parameters fitting by using interaction energy. Fig. 2.8 shows that black dots explain DFT results for H_2-H_2 interaction energy and red line express fitting line. In this H-H fitting, A, B, λ_1, λ_2 and R^e which are obtained from H-H parameters fitting by using binding energy applied to H_2-H_2 parameters fitting and $\alpha,\ \beta,\ \eta,\ \delta$ and c values are presented on Table 2.2 are obtained.

2.5 Results and Discussion

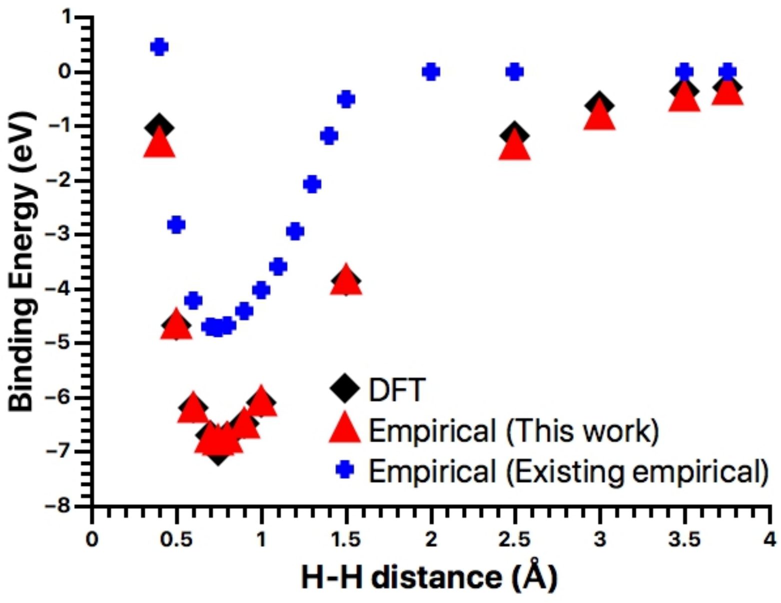


Fig. 2.9. Binding energy after H-H parameters fitting.

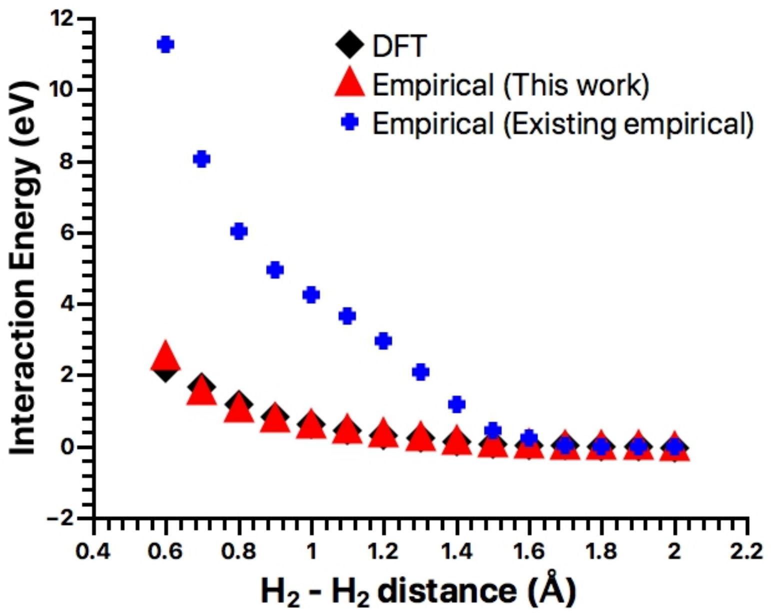


Fig. 2.10. Interaction energy after H-H parameters fitting.

Fig. 2.9 and Fig. 2.10 represent the H-H binding energy and H_2-H_2 interaction energy compared DFT, empirical potential (This work) and existing empirical potential. In Fig. 2.9 and Fig. 2.10, cross data represents the H-H binding energy and H_2-H_2 interaction energy using existing Tersoff empirical potential parameters and important thing is that these data's curve shape is not smooth because in existing classical MD, cutoff range is too short to calculation so it does not calculate for long range interaction. In this study, we fixed cut off range much longer and calculate long range of hydrogen molecule. As we can see, DFT and empirical potential (This work) results match each other after H-H parameters fitting. We have developed a systematic process to construct empirical potential for Si nanowires passivated hydrogen.

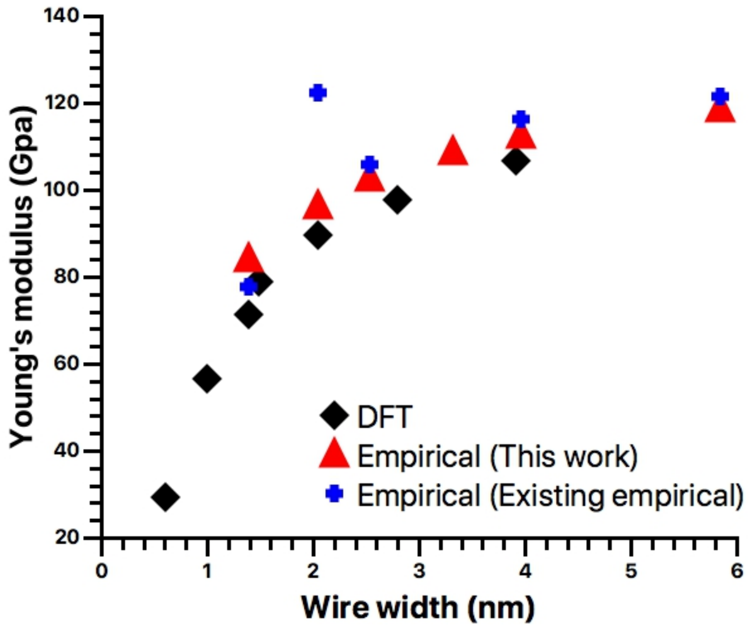


Fig. 2.11. Young's modulus increasing wire width of Si nanowires.

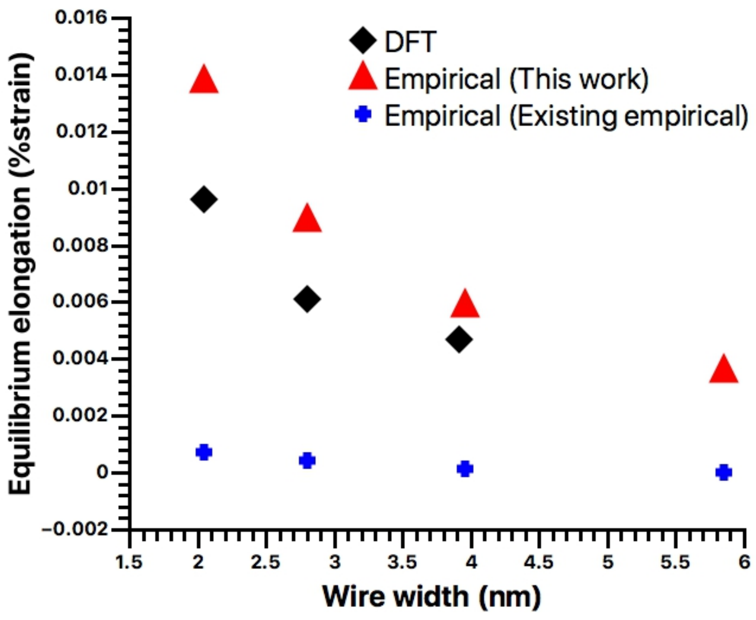


Fig. 2.12. Equilibrium elongation increasing wire width of Si nanowires.

In Fig. 2.11 and Fig. 2.12, mechanical properties were performed by using H-H parameters which are obtained after H-H part parameters fitting. To evaluate the new fit of H-H part, young's modulus and equilibrium elongation of Si nanowires are calculated increasing wire width of Si nanowires and compared with the DFT results and existing empirical results in Fig. 2.11 and 2.12. The size reliance of Young's modulus and equilibrium elongation show critical improvement compared to the DFT results and existing potential results. Until now, surface part of Si nanowires is fitted by using our systematic fitting method and shows mechanical properties to prove enhancement, however, the improvement of the irregular mechanical properties can be seen by the H-H parameter fitting of the surface, but the perfect result matching could not be seen. The reason for this is that not only the surface but also the silicon-hydrogen parameter fitting between the silicon and the surface must be performed. In addition, the hydrogen parameters obtained so far are limited to the calculation of the mechanical properties of nanowires with hydrogen and silicon. Furthermore, it is necessary to derive parameter fittings that can be applied to various types of materials, and potential errors of existing empirical potentials must be corrected for calculations using various materials as well as silicon nanowires.

CHAPTER 3. MD SIMULATIONS OF NANOSIZED CU-PRECIPITATES USING QUENCHING & PARTITIONING (Q&P) PROCESSING

We conducted molecular dynamics based computational study to investigate whether changing the morphologies of Cu precipitates and the ratio of multi-phases in the microstructure can greatly enhance both ultra-high strength and good ductility. In this section, the detailed setup for computational study is listed as follows.

3.1 Modified Embedded Atom Method (MEAM) Potentials

In the MEAM, The total energy function is given as [59]

$$E = \sum_i \left[F_i(\overline{\rho_i}) + \frac{1}{2} \sum_{j \neq i} \phi_{ij} (R_{ij}) \right] \tag{23}$$

where F_i and $\phi_{ij}(R_{ij})$ are the embedding term as a function of the background electron density $\overline{\rho_1}$ and the pair interaction function between atoms i and j is divided by a distance of R_{ij} respectively. For the calculation of the total energy using MEAM potential, the functional forms of F_i and ϕ_{ij} should be defined first. The background electron density is calculated by considering the directionality of bonding. EAM potentials [60] consider solely spherically averaged atomic electron densities, however, MEAM potentials apply to additional angular functions to explain the directional character of bonding. In original version of MEAM [61], only the first nearest-neighbour interactions are taken into consideration. The second and more far nearest-neighbour interactions become crucial while considering many-body screening function [62]. In general, a binary structure chosen by one type of atom has only the same type of atoms as the second-nearest neighbours. The total energy per atom of such structure is calculated using the universal equation of state. After that, the interactions between different kinds of elements are acquired from the embedding energy of the binary structure and the values of the total energy per atom are already obtained.

3.2 The 2NN MEAM Potential Parameters for the Fe – Cu System

In this research, the 2NN MEAM potential parameters for pure Fe and Cu were given from Lee et al. [59].

Table 3.1. The 2NN MEAM potential parameters for Fe and Cu.

	E_c	r_e			$\beta^{(0)}$	$\beta^{(1)}$	$\beta^{(2)}$	$\beta^{(3)}$	$t^{(1)}$	$t^{(2)}$	$t^{(3)}$	C_{min}	C_{max}	
e	.29	.48	.73	.56	.15	.00	.00	.00	.60	.80	7.2	.36	.80	.05
u	.54	.555	.42	.94	.83	.2		.2	.72	.04	.95	.80	.21	.05

The units of the cohesive energy E_C, the distance of equilibrium nearest-neighbor r_e and the bulk modulus B are eV, Å, and 10^{12}dyne/cm^{-2}, respectively. The structure of Fe and Cu is BCC. In Table 3.1, it includes 14 independent potential parameters for pure Fe and Cu in the 2NN MEAM. E_c, r_e, B and d represent the cohesive energy, the distance of equilibrium nearest-neighbour, the bulk modulus and an adjustable parameter respectively. For the electron density part, $\beta^{(0)}$, $\beta^{(1)}$,$\beta^{(2)}$,$\beta^{(3)}$,$t^{(0)}$, $t^{(1)}$, $t^{(2)}$ are the seven potential parameters and A is a potential parameter for the embedding function and C_{min}, C_{max} are for many-body screening. For pure Fe and Cu elements, the three atoms are all of the same type i.e., {i, j, k} = {Fe, Fe, Fe} or {Cu, Cu, Cu}. However, in Fe-Cu alloy systems, four different types i.e., {i, j, k}= {Fe, Cu, Fe}, {Cu, Fe, Cu}, {Fe, Fe, Cu}, and {Fe, Cu, Cu} are employed to represent one of the screening atom and the interaction atoms. The other parameter (ρ_0) is the atomic electron density scaling factor.

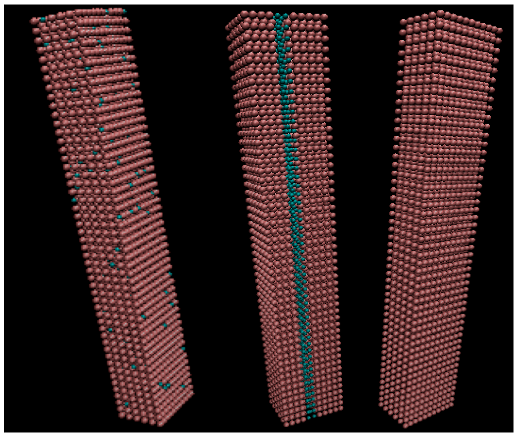
Table 3.2. The 2NN MEAM potential parameters for binary Fe and Cu.

	Fe-Cu	C_{min}(Cu, Fe, Cu)	1.21
E_c	4.2605	C_{min}(Fe, Fe, Cu)	0.7225
r_e	2.57	C_{min}(Fe, Cu, Cu)	0.7225
B	2.72002	C_{max}(Fe, Cu, Fe)	2.8
d	0.05	C_{max}(Cu, Fe, Cu)	2.8
ρ_0	$\rho^{(Fe)}=\rho^{(Cu)}=1$	C_{max}(Fe, Fe, Cu)	2.8
C_{min}(Fe, Cu, Fe)	0.36	C_{max}(Fe, Cu, Cu)	2.8

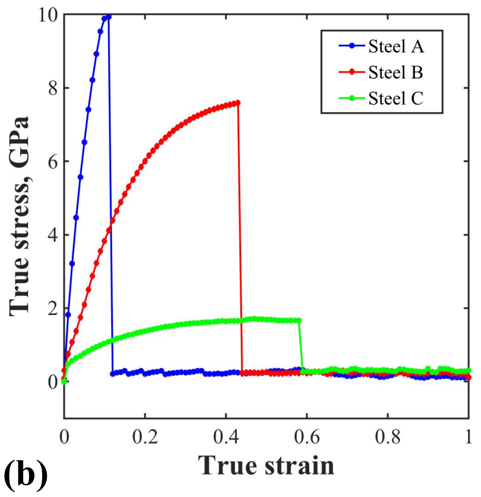
3.3 Molecular Dynamics (MD) Simulations

In simulations, the structures of TRIP, Cu-TRIP with particle Cu precipitates and Cu-TRIP with lamellae Cu precipitates were represented using VESTA (Visualization for Electronic and Structural Analysis) [63]. MD simulations were performed at Purdue Carter Cluster using the software LAMMPS [64]. Fe and Cu were calculated using the Modified Embedded – Atom Method (MEAM) interatomic potential. The energy of the structures was minimized by the conjugate-gradient method with NVE which cannot change the total number of particle (N), the system's volume (V) and the total energy € in the system and designated stopping tolerance for energy (unitless) equals 1.0e-25, stopping tolerance for force (force units) equals 1.0e-25, max iterations of minimizer equals 1,000 and max number of force/energy evaluations equals 10,000. The structure of steel A, steel B and steel C is followed the experimental process by using Q&P process. Steel A was heat treated by using Q&P process (quenching temperature = partitioning temperature) consisting of annealing at 1093.15K for 180000 steps, quenching to 453.15K and partitioning for 300000 steps and then cooled to room temperature (298K). Steel B was treated by using Q&P process (quenching temperature ≠ partitioning temperature) including annealing at 1093.15K for 180000 steps, quenching to 453.15K for 10000 steps and then partitioning at 653.15K for 300000 steps. Finally, the steel B was cooled to room temperature (298K). Steel C except for no Cu-containing was treated by using Q&P process consisting of annealing at 1093.15K for 180000 steps, quenching to 673.15K and partitioning for 300000 steps and then cooled to room temperature (298K).

3.4 The Structure of Binary Fe-Cu System and the True Stress – Strain Curve



(a)



(b)

Fig. 3.1. (a) Three types of initial structure. Steel A: Cu-steel with lamellae Cu precipitates, consisting of 10% Cu in structure; Steel B: Cu-steel with spherical Cu precipitates, including 5% of randomly mixed Cu; Steel C: steel without Cu-containing. (b) True stress-strain curve: Steel A: with lamellae Cu precipitates (10% Cu); Steel B: with spherical Cu precipitates (5% Cu); Steel C: steel without Cu-containing.

In Fig. 3.1. (a), three types of structure (Steel with lamellae Cu precipitates, consisting of 10% Cu in structure; Steel with spherical Cu precipitates, including 5% of randomly mixed Cu; steel without Cu-containing) are used, which include 5866 atoms, in cross-section, with 28.70 Å in width, 20.294 Å in length and 202.94 Å in height for reference structure for all of structure. At interface between Fe and Cu in Cu-steel, lamellae Cu precipitates are considered for the combination of Fe with Cu in (110) direction. For the lamellae structure, 10% of Cu in total atoms is used to construct Cu-steel with lamellae Cu precipitates. For Cu-steel with spherical Cu precipitates, 5% of Cu is mixed in Cu-steel arbitrarily and all of atoms are relaxed for stabilization. Fig. 3.1.(b) represents the true stress–strain curves for two TRIP steels with different Cu precipitates (lamellae, spherical) compared with the steel without Cu precipitates. In simulation, Cu-steel with lamellae and spherical Cu precipitates shows evidently higher strengths than that of the TRIP steel without Cu precipitates. In linear elastic regime, For Cu-steel with lamellae Cu precipitates, the tensile yield strength σ_y (at 0.2% offset) reaches as high as 6.52 Gpa and the ultimate tensile strength (σ_{UTS}) is 9.93 Gpa. For Cu-TRIP with particle Cu precipitates, 4.18 Gpa and 7.58 Gpa are observed for σ_y and σ_{UTS}, respectively. In contrast, the steel without Cu shows the relatively low σ_y of 0.48 Gpa and σ_{UTS} of 1.69 Gpa. The experimental finding is validated by the molecular dynamical simulations, which confirmed that changing the morphologies of Cu precipitates and the ratio of multi-phases in the microstructure can greatly enhance both ultra-high strength and good ductility.

CHAPTER 4. MD SIMULATION EXPLORING DEFORMATION BEHAVIOUR OF NANOSIZED FE3C STRENGTHENED UFG FERRITIC STEEL

MD simulation based on MEAM was employed to explain the strengthening effect of nanosized Fe3C precipitates in Ferrite-Fe3C system. Both size and volume faction of Fe3C particles play important role for improving the strength of UFG ferritic steel, i.e., the strength of UFG D6AC steel increases with increasing volume fraction and decreasing size of Fe3C particles. In this section, the detailed setup for computational study is listed as follows.

4.1 The 2NN MEAM Potential Parameters for the Fe – C System

In this research, the 2NN MEAM potential parameters for pure Fe and C were given from Lee et al. [65].

Table 4.1. The 2NN MEAM potential parameters for Fe and C. The units of the cohesive energy E_C, the distance of equilibrium nearest-neighbor r_e and bulk modulus B are eV, Å, and 10^{12}dyne/cm^{-2}, respectively. The structure of Fe and C are BCC and diamond.

	E_c	r_e	B	A	$\beta^{(0)}$	$\beta^{(1)}$	$\beta^{(2)}$	$\beta^{(3)}$	$t^{(1)}$	$t^{(2)}$	$t^{(3)}$	C_{min}	C_{max}	d
Fe	4.29	2.48	1.73	0.56	4.15	1.00	1.00	1.00	2.60	1.80	-7.2	0.36	2.80	0.05
C	7.37	1.54	4.45	1.18	4.25	2.8	2.0	5.0	3.2	1.44	-4.48	2.80	1.41	0.00

In Table 4.1, it includes 14 independent potential parameters for pure Fe and C each in the 2NN MEAM. E_c, r_e, B and d represent cohesive energy, the distance of equilibrium nearest-neighbor, bulk modulus and an adjustable parameter respectively. For the electron density part, there are seven potential parameters that are $\beta^{(0)}$, $\beta^{(1)}$, $\beta^{(2)}$, $\beta^{(3)}$, $t^{(0)}$, $t^{(1)}$, $t^{(2)}$ and A is potential parameter for the embedding function and C_{min}, C_{max} for many-body screening.

Table 4.2. The 2NN MEAM potential parameters for binary Fe and C. The units of the cohesive energy E_C, the distance of equilibrium nearest-neighbor r_e and bulk modulus B are eV, Å, and 10^{12}dyne/cm^{-2}, respectively. The structure of Fe and C are BCC and diamond.

	Fe-C
E_c	6.01
r_e	2.364
B	2.644
d	0.05
ρ_0	$\rho^{(c)}/\rho^{(Fe)}$=6
C_{min}(Fe, C, Fe)	0.36
C_{min}(C, Fe, C)	0.16
C_{min}(Fe, Fe, C)	0.16
C_{min}(Fe, C, C)	0.16
C_{max}(Fe, C, Fe)	2.8
C_{max}(C, Fe, C)	1.44
C_{max}(Fe, Fe, C)	2.8
C_{max}(Fe, C, C)	2.8

As I mentioned above, two parameters (C_{min}, C_{max}) must be determined to explain Ferrite-Fe3C systems. The first value of determined parameter is C_{min}. In Table 4.1, Fe and C elements has its own value of C_{min} each. C_{min} decides the extent of screening of an atom k to the interaction with two neighbor atoms (i, j atoms). For pure Fe and C elements, the three atoms are all the same type i.e., {i, j, k} = {Fe, Fe, Fe} or {C, C, C}. However, in Ferrite-Fe3C systems, one of the screening atom and the interaction atoms should be four different types i.e., {i, j, k} = {Fe, C, Fe}, {C, Fe, C}, {Fe, Fe, C}, and {Fe, C, C}. The other parameter (ρ_0) is the atomic electron density scaling factor. In equilibrium structure, if R equals to r_e, the value of all atomic electron density becomes ρ_0 which is a random value and cannot effect on calculations for Fe and C pure elements. On the other hand, if the constituted elements have different coordination numbers in systems, the scaling factor relative difference great influence on calculations. In Table 4.2, the 2NN MEAM potential parameters (E_c, r_e, B, d) for Fe-C were given from Lee et al. [65] by using their procedure for the parameters determination.

4.2 Molecular Dynamics Simulations

In simulations, structures of UFG1 and UFG2, were represented by using VESTA (Visualization for Electronic and Structural Analysis) [63]. MD simulations were performed at Purdue Rice Cluster using the software LAMMPS [64]. Fe and C were calculated by using Modified Embedded – Atom Method (MEAM) interatomic potential. The energy of the structures was minimized by the conjugate-gradient method and the total number of particles in the system (N), the volume of system (V) and the total energy in the system (E) are constant quantities. This is called the NVE ensemble which is used in this simulation then designated stopping tolerance for energy (unitless) equals 1.0e-25, stopping tolerance for force (force units) equals 1.0e-25, max iterations of minimizer equal 5000 and max number of force/energy evaluations equals 10000.

4.3 The Stress – Strain Relationship in Molecular Dynamics Simulations

In the stress-strain relationship, the structure of simulation is defined using the constant total number of particles, constant-energy, constant-volume ensemble (NVE) is pulled in the y-direction, or perpendicular to the boundary interface, to increase strain. The strain in increased for a specified number of times in a loop, and the stress is calculated at each point before the simulation loops. The stress-strain curve can be generated using MATLAB script.

4.4 The Structure of Ferrite and Fe3C

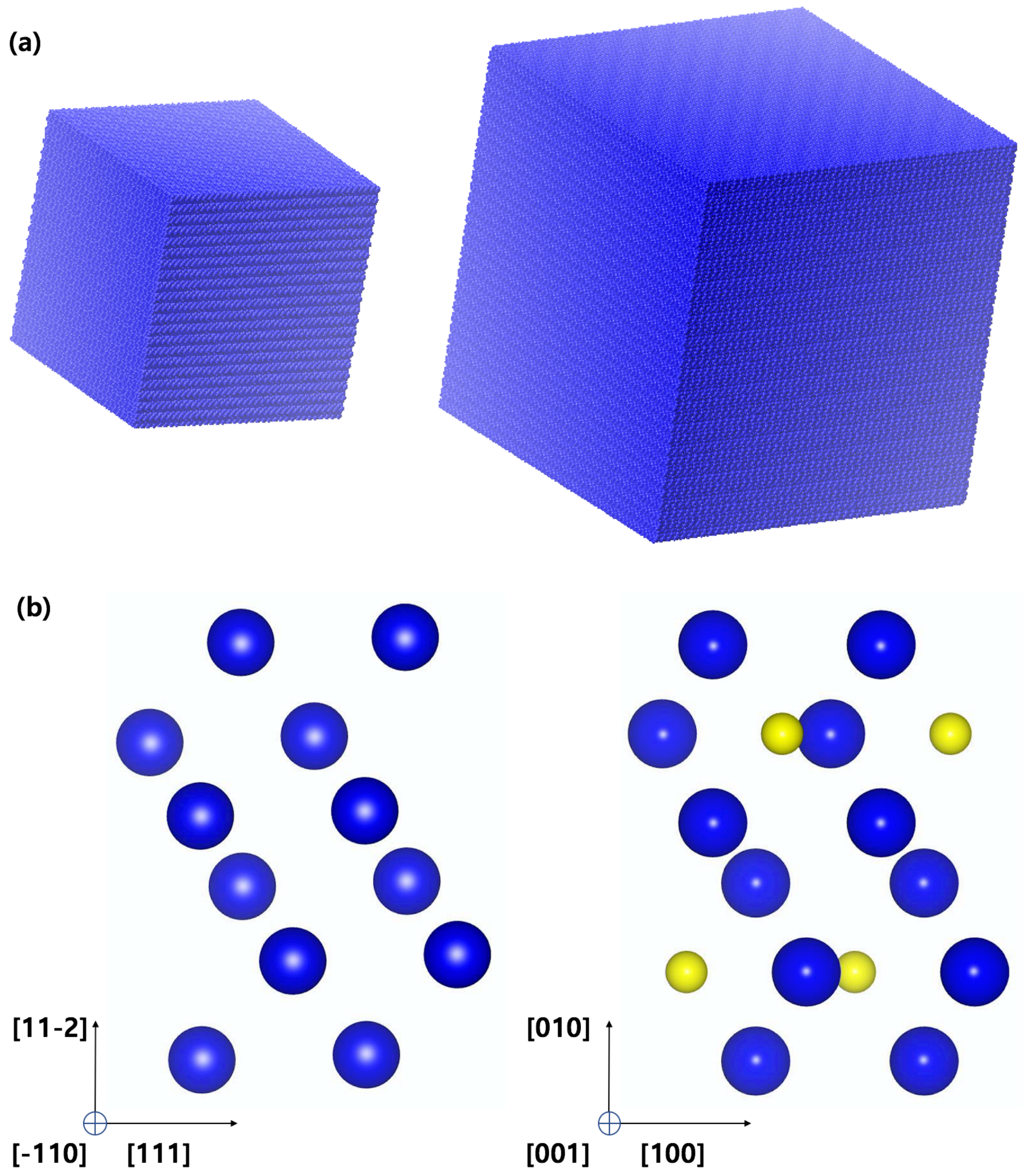


Fig. 4.1. Two types of structure for ferrite and Fe3C. (a) UFG1 which includes 9.6% of cementite in ferrite (left) and UFG2 which includes 5.5% of cementite in ferrite (right). (b) The interface orientation relationship between ferrite and Fe3C is used $[111]_f \parallel [100]_c$, $[-110]_f \parallel [001]_c$ and $[11-2]_f \parallel [010]_c$.

In order to conduct the atomistic simulation, two types of structure (UFG1, UFG2) are used which includes 268980 and 1032496 atoms respectively in Fig. 4.1. First, the dimension of the UFG1 is 14.3nm and the size of the cementite which includes in UFG1 is 2.3nm. Second, the dimension of the UFG2 is 22.6nm and the size of the cementite which includes in UFG2 is 3nm. The interface orientation relationship between ferrite and Fe3C is used $[111]_f \parallel [100]_C$, $[-110]_f \parallel [001]_c$ and $[11-2]_f \parallel [010]_c$ by Bagaryatsky [66].

4.5 Results

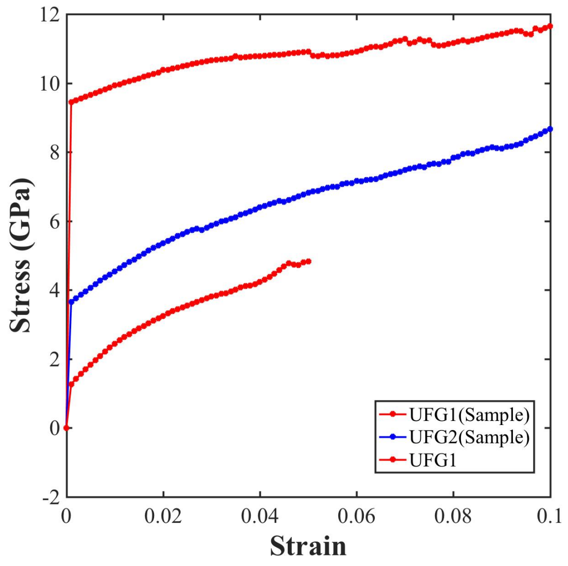


Fig. 4.2. True stress-strain curve: (a) UFG1 (Sample), (b) UFG2 (Sample) (c) UFG1.

Fig. 4.2 represents true stress–strain curves for UFG1 (Ferrite + 9.6% of Fe3C) and UFG2 (Ferrite + 5.5% of Fe3C). In simulation, First, for the stress-strain curve, UFG1 (Sample) and UFG2 (Sample) which are diminished almost 100times and 608 and 1796 atoms are used respectively in this simulation. Sample dimensions of UFG1 and UFG2 are 3.40076nm by 4.77166nm in the x-z plane and 0.4484 nm and 1.08408nm in the y axis respectively. UFG1 (Sample) and UFG2 (Sample) show that UFG1 (Sample) is evidently higher strengths than UFG2 (Sample). In linear elastic, For UFG1 (Sample), the tensile yield strength σy (at 0.2% offset) reaches as high as 9.3 Gpa and the ultimate tensile strength (σ_{UTS}) is 11.3 Gpa. In contrast, UFG2 (Sample) reveals the σ_y of 3.8 Gpa and σ_{UTS} of 8.5 Gpa. Finally, UFG1 which are diminished almost 30times and 268980 atoms are used in this simulation. Dimension of UFG1 is 14.3nm and the size of the cementite which includes in UFG1 is 2.3nm. In linear elastic, the tensile yield strength σ_y (at 0.2% offset) reaches as high as 1.2 Gpa and σ_{UTS} of 4.2 Gpa.

CHAPTER 5. THE PERIDYNAMICS MODEL

The peridynamics is the non-local extension of the classical continuum mechanics. The model structure of peridynamics is the same as a MD model and in LAMMPS, SI units can be used for simulation. Compared with FEM (Finite Element Method), FEM can only acquire approximate solutions and the user's mistake in obtaining a solution can be incredibly deadly, and the results can vary depending on how the mesh is set. Finally, the FEM is based on partial differential equations, but there are no partial derivatives on crack surfaces. Therefore, to secure this weakness, the peridynamics model is used. The peridynamics model is based on integral equations. In the analysis of damage and cracks, the integral equations of the peridynamics theory can be apply directly, because they do not require partial derivatives. Accordingly, it overcomes deficiencies in modeling of deformation discontinuities. Next, let's look at the govern equations of the peridynamics model [25-28].

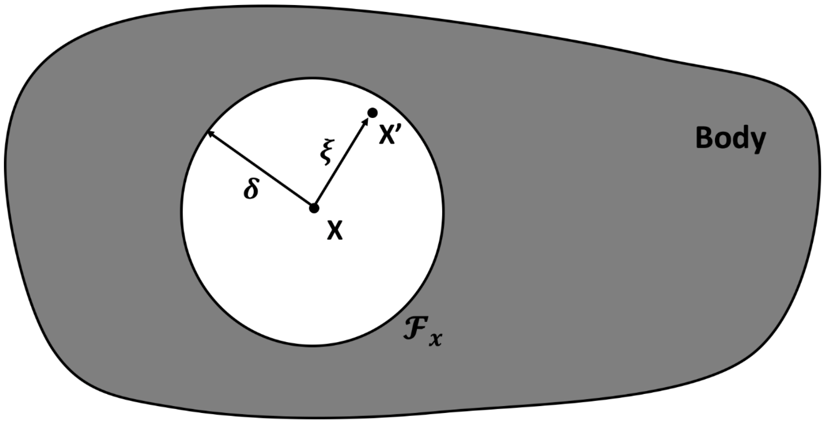


Fig. 5.1. The composition of peridynamics model.

Fig. 5.1 is plotted to represent the composition of the peridynamics model. As you can see in the Fig. 5.1, $\mathcal{F}_x$ (Family of x) exists in the body, and there are arbitrary points x and x' in it. X and x 'are the points given in the reference configuration. U(x,t) and y(x,t) represent the displacement and position of the point x at time t respectively. In peridynamics theory, all points, including x and x' in the horizon δ boundary, are interconnected by bonds. That is, each point has connectivity to all points in horizon δ as well as nearby neighbors. Thus, as mentioned above, the peridynamics is a non-local extension of the classical continuum mechanics. Integral equation is used to calculate the current point forces per unit volume. Due to spatial differentiation is not used in peridynamics, it is useful for analyzing discontinuous media and discrete particles such as cracks or damage. The peridynamic equation of motion is given as:

$$\rho(x)\ddot{u}(x,t) = \int_{\mathcal{F}_x} \omega(u(x',t) - u(x,t), x' - x) dV_{x'} + b(x,t), \qquad t \geq 0 \tag{24}$$

$\rho(x)$ is the density in the reference configuration, $\mathcal{F}_x$ is the family of x in the horizon δ, b represents the external force per unit volume and u represent the displacement of the point x at time t. ω is the pairwise bond force density that includes all of the information related to x and x'. Next, the relative position vector which is time-independent and displacement vector which is time-dependent of two bonded points x and x' are defined by $\xi = x' - x$ and $\eta = u(x', t) - u(x, t)$, respectively. ξ + n represents the current relative position vector between the particles. The pairwise bond force density ω should have the following properties:

$$\omega(-\eta, -\xi) = -\omega(\eta, \xi), \qquad (\xi + \eta) \times \omega(\eta, \xi) = 0 \ \ \forall \eta, \xi \tag{25}$$

$\omega(-\eta, -\xi) = -\omega(\eta, \xi)$ represents the conservation of linear momentum, and $(\xi + \eta) \times \omega(\eta, \xi) = 0$ indicates the conservation of angular momentum. It means that the force vector between x and x' is parallel to their current relative position vector.

5.1 Prototype Microelastic Brittle (PMB) Model

In this study, crack pattern generated by impact of indenter on disk is used as input data of machine learning algorithm. Therefore, Prototype Microelastic Brittle (PMB) model [27] was used for constitutive model to obtain a more precise and complex crack pattern. The govern equations are given as:

$$s = \frac{|\xi + \eta| - |\xi|}{|\xi|} = \frac{y - |\xi|}{|\xi|} \tag{26}$$

f is the scalar valued bond force and it is defined in relation to bond stretch s. The best way to apply failure to the constitutive model is to allow the bonds to break when they exceed the predefined limit. If the bond is broken, the tensile strength cannot be restored. The PMB model is defined as follows.

$$f(y(t),\xi) = g\big(s(t,\xi)\big)\mu(t,\xi) \tag{27}$$

$$g(s) = cs \quad \forall s \tag{28}$$

$$\mu(t,\xi) = \begin{cases} 1 & if\ s(t',\xi) < s_0 \quad for\ all\ \ 0 \le t' \le t \\ 0 & otherwise \end{cases} \tag{29}$$

f is composed of the products of g and μ functions. g is the linear scalar-valued function which is composed of spring constant (c) and bond stretch (s). μ is a history-dependent scalar-valued function and has a value of 1 and 0 depending on the condition. In other words, if the bond stretch is less than critical bond stretch (s_0), it has a value of 1, otherwise it has a value of 0. The spring constant (c) and the critical bond stretch (s_0) mentioned above are described in detail below.

$$c = \frac{18k}{\pi\delta^4} \tag{30}$$

$$s_0 = \sqrt{\frac{10G_0}{\pi c\delta^5}} = \sqrt{\frac{5G_0}{9k\delta}}\ , \quad G_0 = \frac{\pi c s_0^2 \delta^5}{10} \tag{31}$$

k is the bulk modulus of the material and G_0 is the work required to break all bonds per unit fracture area. As we can be seen from the above equations, the spring constant and the critical bond stretch are composed of bulk modulus of the material (k) and chosen horizon boundary (δ). Finally, in the case of brittle materials such as glass, related to bond stretch, is defined as follows.

$$s_0 = s_{00} - \alpha s_{min}(t), \quad s_{min}(t) = \min_{\xi}\left\{\frac{y(t) - |\xi|}{|\xi|}\right\} \tag{32}$$

s_{min} is the current minimum stretch in the group of all bonds connected to a given point, s00 and α are constants and α is generally used about 1/4.

5.2 Data Preparation

In this study, crack pattern data were generated by using the peridynamics in the LAMMPS, a kind of MD simulation tool.

5.2.1 LAMMPS Input File Setup

The spherical indenter impacted the disk by dropping the spherical indenter in the direction perpendicular to the disk, it means from the + y axis to disk and varying the x and z coordinates of the indenter to impact whole parts of the disk. The radius of the indenter are 0.007 m and 0.008 m for the inverse problem and the velocity are 100m/s and 100.1m/s, however, for the forward problem, the radius of the indenter and velocity are fixed by 0.0020m and 100m/s respectively. The radius of the cylindrical disk is 0.037 m, the thickness is 0.0025 m and this disk contains 12855 particles. Each particle i has volume fraction $V_i = 1.25 \times 10^{-10} m^3$ and the density of the disk material is $\rho = 2200 kg/m^3$. The pair style in LAMMPS is used peri / pmb [27] model. Pair constants are c, horizon, s_{00} and α in order of 1.6863e22, 0.0015001, 0.0005, 0.25. c is the spring constant for peridynamic bonds, the horizon is a cutoff distance and s00 and alpha are used as a critical bond stretch parameters.

5.2.2 LAMMPS input .peri file

Table 5.1. Algorithm of .peri input file.

Algorithm: LAMMPS .peri input file

```
1: # Define number of iteration steps, l = Velocity, k = Indenter radius, i, j = Hitting location
2: variable imax equal 100
3: variable jmax equal 100
4: variable kmax equal 0
5: variable lmax equal 0
6: variable countmax equal "v_imax*v_jmax*v_kmax*v_lmax"
7: # Start loops
```

Table 5.1 continued

```
label start_of_loop_1
variable l loop ${lmax}
label start_of_loop_2
variable k loop ${kmax}
label start_of_loop_3
variable i loop ${imax}
label start_of_loop_4
variable j loop ${jmax}
clear
# Define initial condition
units                si
boundary       s s s
dimension 3
atom_style      peri
atom_modify        map array
neighbor       0.0010 bin
# Target (disk)
lattice        sc 0.001
region         target cylinder y 0.0 0.0 0.037 -0.0025 0.0 units box
create_box      1 target
create_atoms    1 region target
pair_style      peri/pmb
pair_coeff      * * 1.6863e22 0.0015001 0.0005 0.25
set            group all density 2200
set            group all volume 1.25e-10
velocity       all set 0.0 0.0 0.0 sum no units box
fix            1 all nve
fix            2 all indent 1e16 sphere v_x v_y v_z v_w units box
# Indenter velocity control(q), Indenter radius (w), Disk radius(r) for indent.txt file
variable        q equal "1+(0.001*v_l)"
```

Table 5.1 continued

```
variable        w equal "0.0020+(0.001*v_k)"
variable r equal 0.037
variable       y0 equal 0.00510
variable       vy equal -100
variable       y equal "v_y0 + v_q*step*dt*v_vy"
#Hit the disk reduced 5times compare to disk size
variable       x equal "cos((2*PI)*v_i/v_imax)*v_r*(0.2)"
variable       z equal "sin((2*PI)*v_j/v_jmax)*v_r*(0.2)"
compute         1 all damage/atom
timestep       1.0e-7
thermo         100
dump           ${count} all custom 1000 Data/dump_${count}.peri id type x y z c_1
run           2000
# save indenter.txt for machine learning process
print $x file Data/indenter_${count}.txt
print $z append Data/indenter_${count}.txt
print $w append Data/indenter_${count}.txt
print $q append Data/indenter_${count}.txt
next j
next count
jump SELF start_of_loop_4
next i
jump SELF start_of_loop_3
next k
jump SELF start_of_loop_2
next l
jump SELF start_of_loop_1
```

LAMMPS MD simulation tool are used to obtain the crack pattern data generated by the indenter using peridynamics, and the .peri input file was specified in Table 5.1. We can obtain the specific data by changing the variable using for loops to adjust the velocity and radius of the indenter and the hitting point of the disk. SI units are to be used and boundary is non-periodic and shrink-wrapped. The particle volume is used a simple cubic lattice and the lattice constant is 0.001m. In the line 26, cylinder disk (target) with a radius of 0.037 m and a thickness of 0.0025 m is set. The initial velocity of all particles on line 33 is 0, and line 34 instructs LAMMPS to integrate time with velocity-verlet. The hitting location, velocity, and radius of the indenter should be saved to do damage prediction for the next steps after the MD simulation, so we set up to save the parameters information in indenter.txt. In addition, for the inverse problem, damage is given to the whole parts of the target, but in the forward problem, peridynamics proved to be inaccurate in the boundary part, and data is obtained by only damaging 1/5 size around the center of the target. Finally, we declare a dump command to obtain the crack pattern image, and 2000 time-steps of running simulation to get the data are used, and we set it to 2000 time-steps in dump command as well. The reason for this is that the image we want for prediction in next step is the initial image and the final image due to the same as the meaning of 2000 in the dump command is to store the x,z coordinates of the atoms at the initial target and 2000 time-steps of the target.

5.3 Data Results

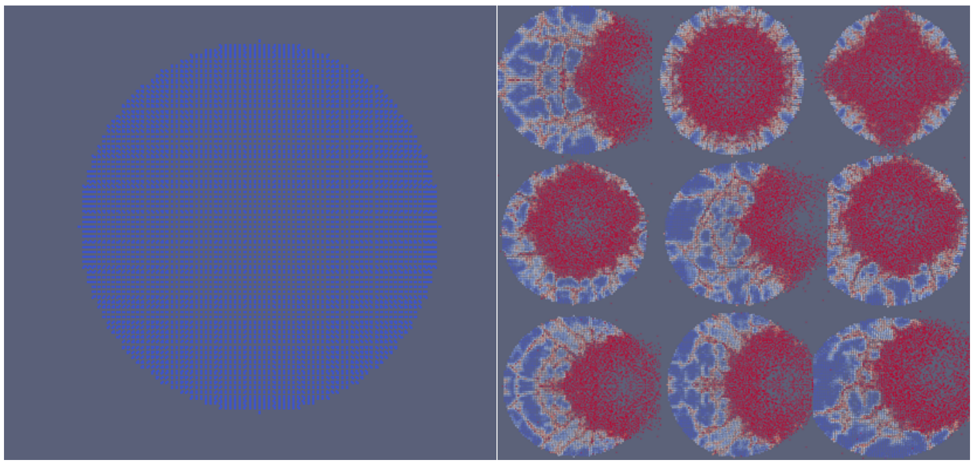


Fig. 5.2. The initial appearance of the disk (Left) and the data result after hitting the indenter onto the disk (Right).

Using the above-mentioned algorithm, we can see the result obtained by simulation using LAMMPS tool in Fig. 5.2. The dump file after running the MD simulation is converted to an EnSight data format with the pizza.py toolkit [67] and Paraview is used for visualization in Fig.5.2 [68]. The left figure is the initial model of the target disk, and the right is the figure after the simulation results. The symmetry solution represents the right side in the figure because of the perfect lattice is used, and the perfectly spherical indenter impacted geometric center of the target. In this study, the use of asymmetric crack patterns could be better data for real life applications but focused primarily on crack patterns with symmetry, However, as the position of the hitting point changes, various crack patterns can be confirmed, so that the data for machine learning is sufficient.

CHAPTER 6. DATA-DRIVEN DISK DAMAGE PREDICTION USING DEEP MACHINE LEARNING

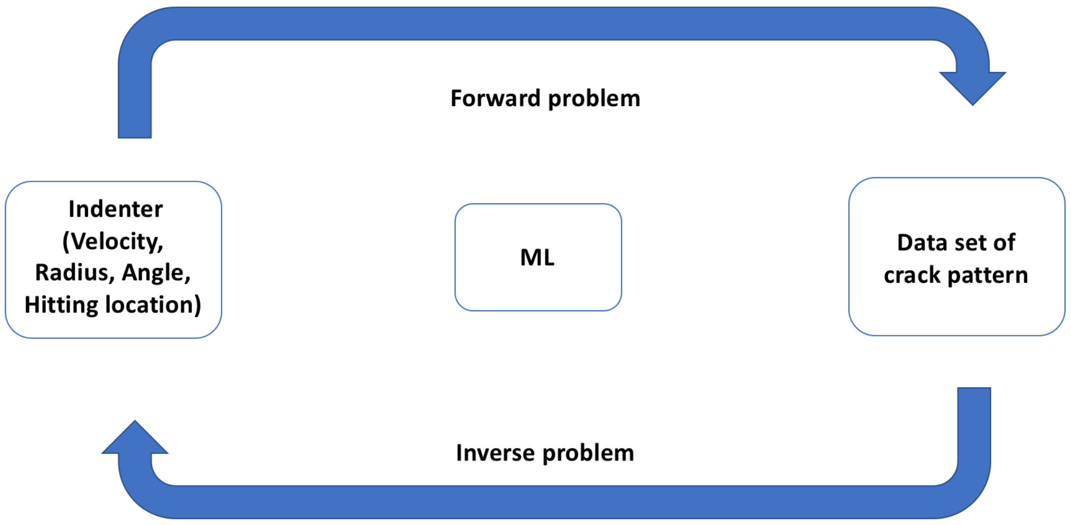

Forward problem
Indenter (Velocity, Radius, Angle, Hitting location)
ML
Data set of crack pattern
Inverse problem

Fig. 6.1. Diagram of inverse problem and forward problem using machine learning.

In this study, we used data obtained by using peridynamics to predict the forward and inverse problems through machine learning and the diagrams are expressed in Fig. 6.1. For the forward problem, the crack patterns of the target disk are predicted due to the hitting locations. Conversely, we use the crack pattern in the inverse problem to predict the velocity, radius, and hitting locations of the indenter.

6.1 Supervised Machine Learning for Inverse Problem

Clarified input and output data set are required to apply the supervised machine learning in this experiment. Various types of data are available to be used as an input data. Moreover, the features of the input data are determined, and its usually expressed as the vector form. In this experiment, image form of damage data, which is simulated by LAMMPS, is used as an input data set. Let the damage data be $\chi = \{\chi_1, \chi_2, \ldots, \chi_N\}$ as an input data when $N \in \mathbb{R}$. For Supervised Learning, labels which classify and represent the input data are required as well. Consider the label set is $\omega = \{\omega_1, \omega_2, \ldots, \omega_M\}$ in case $M \in \mathbb{R}$, and, each label is assigned to the corresponding input data, χ. The labeled data set, which is called the training pair, is $\rho = (\chi_i, \omega_i)$, so this pair represents the damage set with corresponding labels. In this article, 70% of the input data set is used for the training set, and 30% of the input data, is used for the validation set. This labeled training set is sent to a supervised learning algorithm which is chosen and designed for making a prediction model. The goal of the supervised machine learning algorithm is to efficiently classify the training set according to the corresponding labels. There are a few supervised learning algorithms which are used such as decision trees, Naive Bayes, Artificial Neural Network(ANNs), etc [69]. In this article, Convolutional Neural Network(CNNs), is used as Supervised Learning algorithm. Through the learning algorithm, the labeled training set and validation set are used as an input of a function which allows the learning algorithm to classify the relationship between the unlabeled input data, χ, and the corresponding labels, ω. Thus, it generates a function that is $\mathcal{F}: \chi_i \rightarrow \omega_i$. Moreover, through the validation set which is sorted from the input data, the function is optimized as a classifier, and it makes an accurate prediction model. In this process, the more training data is used, the more accurate prediction model is able to be made. Lastly, in order to estimate the prediction model, a new input data set is used as a test set. In the same way as the training set, the corresponding labels is assigned to the new test set. This test set is sent to the prediction model, and it finally shows the accuracy of the prediction model by predicting the assigned labels of the test set. In this process, the labels of the test set are not used for training but for prediction. Thus, the prediction model predicts the corresponding labels of each component of the data set, and the predicted labels are compared with the desired labels of the test pairs. In this article, the result is displayed as a success rate of prediction of the test set, and it shows the accuracy of this supervised learning. An inverse problem is a mathematical way to predict the parameters through the measured and observed data. In this study, the crack pattern on a disk by an indenter was supposed to be analyzed depending on different value of the parameters of the indenter through the way of inverse problem. The purpose of this experiment is by using the concept of the inverse problem for the crack pattern to predict the velocity, radius and hitting points of indenter. A crack patterns of the target data are switched dump file data to Numpy array data file.

6.1.1 Input Image Data for Machine Learning

Table 6.1. Setup for 8 mode input image data.

Mode	4 variables Radius of indenter (r) = 0.007, 0.008m Velocity of indenter (v) = 100, 100.1m/s Angle of indenter (a) = 0°,45° Hitting location of disk (x, y)
Mode1	r=0.007m, v=100m/s, 0°
Mode2	r=0.007m, v=100.1m/s, 0°
Mode3	r=0.008m, v=100m/s, 0°
Mode4	r=0.008m, v=100.1m/s, 0°
Mode5	r=0.007m, v=100m/s, 45°
Mode6	r=0.007m, v=100.1m/s, 45°
Mode7	r=0.008m, v=100m/s, 45°
Mode8	r=0.008m, v=100.1m/s, 45°

Matconvnet [70] based on MATLAB CODE was applied to the inverse problem using Numpy array as input image data. Prior to considering the results, focusing on the input image data by using LAMMPS, we can classify into 8 modes changing the 4 variables (Radius of indenter, Velocity of indenter, angle of indenter, and hitting location of disk) and obtained 7,200 training data sets and 1,200 test data sets. The mode according to the variation of the variable is set as a combination of velocity, angle, radius of indenter and hitting location of disk and is shown in Table 6.1.

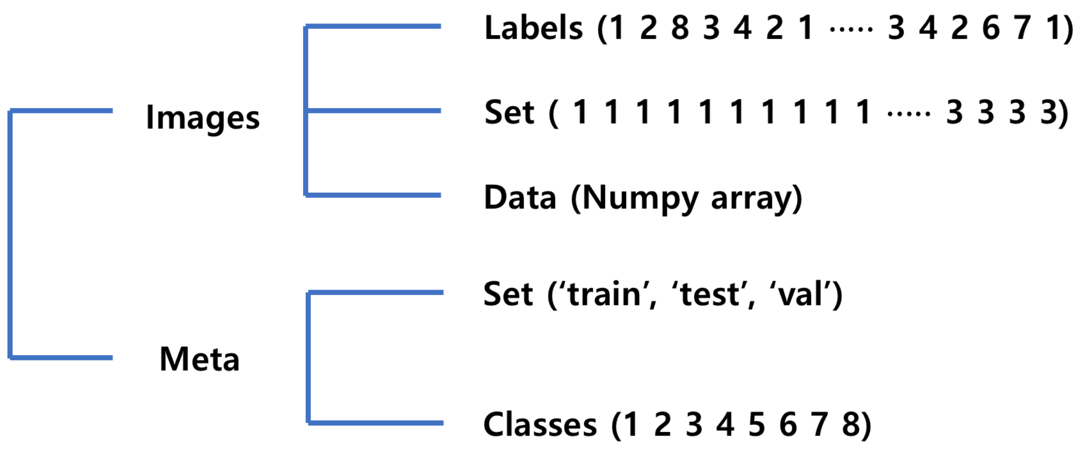

Images
Labels (1 2 8 3 4 2 1 ····· 3 4 2 6 7 1)
Set (1 1 1 1 1 1 1 1 1 1 ····· 3 3 3 3)
Data (Numpy array)
Meta
Set ('train', 'test', 'val')
Classes (1 2 3 4 5 6 7 8)

Fig. 6.2. Data structure in the .mat file.

Next, we distinguished training data set and test data set to apply input image data set to machine learning. First, the image size of the training data set is 128 by 128, and the number is 7200. 2,160 (30% of 7,200 training data set) were designated as validation set. In this way, the image size of the test data set is 128 by 128, and the number of images is 1200. 360 (30% of 1,200 test data set) were designated as validation set. Finally, the training data set and the test data set are made into a .mat file that can be applied to Matconvnet and its structure is shown in Fig. 6.2. The structure of the data in the .mat file is largely divided into 'images' and 'meta'. There are 'labels', 'set', and 'data' classes in the 'images' class. 'label' displays labels on images of 7200 training data sets. Labels were displayed randomly from 1 to 8 since data has 8modes. In the set class, training data, test data, and validation data are indicated by 1, 2, and 3, respectively. The data class converts the data obtained from the LAMMPS into a Numpy array and expresses it as a 4-D single matrix. Finally, there are 'set' and 'classes' in the meta class. The 'set' contains the words 'train', 'test', and 'val' which are related to the 'images' class and indicate that the 1 2 3 in the set of images class is training data, test data, and validation data, respectively. The 'class' in the 'meta' specifies a class from 1 to 8 due to this inverse problem has a mode of 8.

6.1.2 Numerical Results

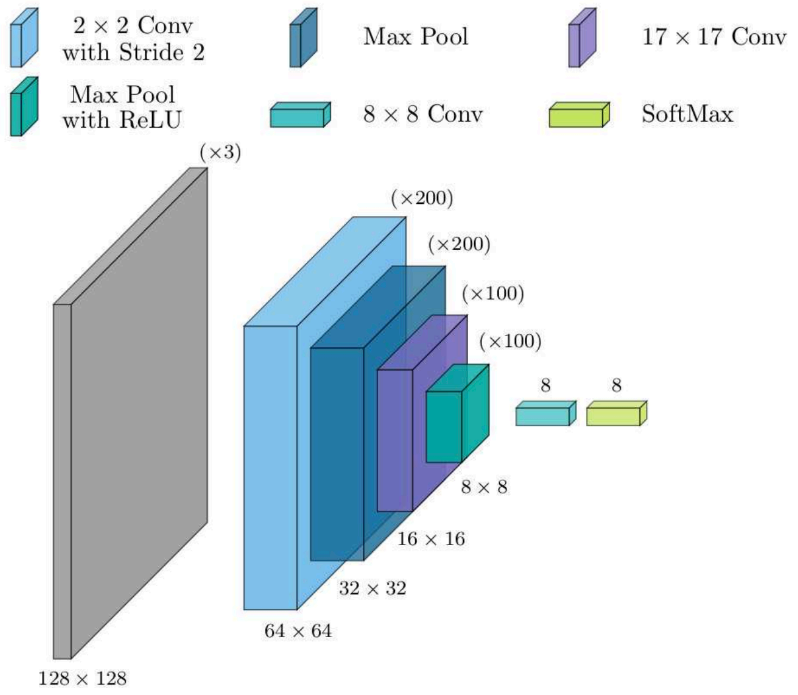


Fig. 6.3. Architecture of convolution neural network (CNN) for inverse problem.

Data was trained by Matconvnet based on MATLAB using training data set and test data set. Convolution neural network (CNN) was used for training and its structure was shown in Fig. 6.3. Our fully convolutional network is composed of two convolutional layers followed by a max pool layer and rectified linear layer (ReLU) and followed by fully-connected layer with softmax layer which is used in the final layer of convolution neural network-based classifier. In the first convolutional layer, the "weights" are the filters being learned and initialized with random numbers from a Gaussian distribution. The filters are 2 by 2 spatial resolution with 1filter depth and number of kernels are 200. The next layer is a max pooling layer which takes 2 by 2 sliding window with a stride of 2. The spatial resolution will be decreased due to the stride 2 in max pooling layer. The next layer is the rectified linear unit (ReLU). It promotes the convergence of stochastic gradient descent through the negative value becomes zero. By doing this activation, the enhancement of the speed of convergence can be seen when training the data. The last layer is fully-connected layer have full connections with the activation functions in the previous convolution layers. In this study, the output of this layer *must* be 1x1 spatial resolution and since the size of the filter should be classified as 8 mode, it should be unconditionally 8.

Table 6.2. Results of CNN prediction for inverse problem.

Test image number : Label (Mode)	CNN prediction results (Mode)
Test image 1 : 5 (Mode)	5 (Mode)
Test image 100 : 6 (Mode)	6 (Mode)
Test image 200 : 7 (Mode)	7 (Mode)
Test image 300 : 2 (Mode)	2 (Mode)
Test image 400 : 8 (Mode)	8 (Mode)
Test image 500 : 7 (Mode)	7 (Mode)
Test image 600 : 2 (Mode)	1 (Mode)
Test image 700 : 3 (Mode)	3 (Mode)
Test image 800 : 8 (Mode)	8 (Mode)
Test image 900 : 6 (Mode)	6 (Mode)
Test image 1000 : 7 (Mode)	7 (Mode)
Test image 1100 : 7 (Mode)	7 (Mode)
Test image 1200 : 5 (Mode)	5 (Mode)

Using the CNN mentioned above with 7200 training data and 1200 test images, the learning rate is 0.001 and the epoch is trained for 25, and the results are shown in Table 6.2. As you can see in the table, the left side shows the number of the 13 test images and the marked label among 1200 test images, and the right side shows the prediction based on the training results. The prediction results using the training data showed a success rate of 95.6%.

6.2 Forward Problem

The forward problem consists of predicting the crack pattern which is produced by an indenter striking the disk at a specified location. More precisely, given the x-z coordinates of the center of the indenter's hit location we aim to predict the damage pattern across the disk. The dataset for the forward problem has been created using the LAMMPS peridynamics package. Simulations corresponding to 25,250 indenter hit locations were carried out to construct the final dataset. To avoid the introduction of potentially unrealistic artifacts in the dataset due to inaccuracies of the peridynamics model near the boundary, the selected hitting locations have been restricted to the inner 1/3 of the target disk. In order to successfully train the network for the forward problem, it was necessary to design a loss function which converted the raw LAMMPS damage data into a format which is compatible with the deep learning framework and array/tensor-based format of the TensorFlow network implementation. This conversion was performed most naturally by bucketizing the atoms into a uniform grid of resolution 64 × 64. The average damage of the atoms contained in each bucket/bin of the grid was calculated and used to create the target outputs for the network. It was also necessary to account for atoms which had been substantially dislocated by a direct impact with the indenter and had left some bins at the disk's interior empty. These interior bins were assigned a full damage value of 1.0 to indicate that the atom which had originally occupied the space was displaced by the indenter. This was achieved by storing a copy of the LAMMPS data before the indenter strikes to serve as a template which indicates where the atom damages should be calculated.

6.2.1 Network Structure for the Forward Problem

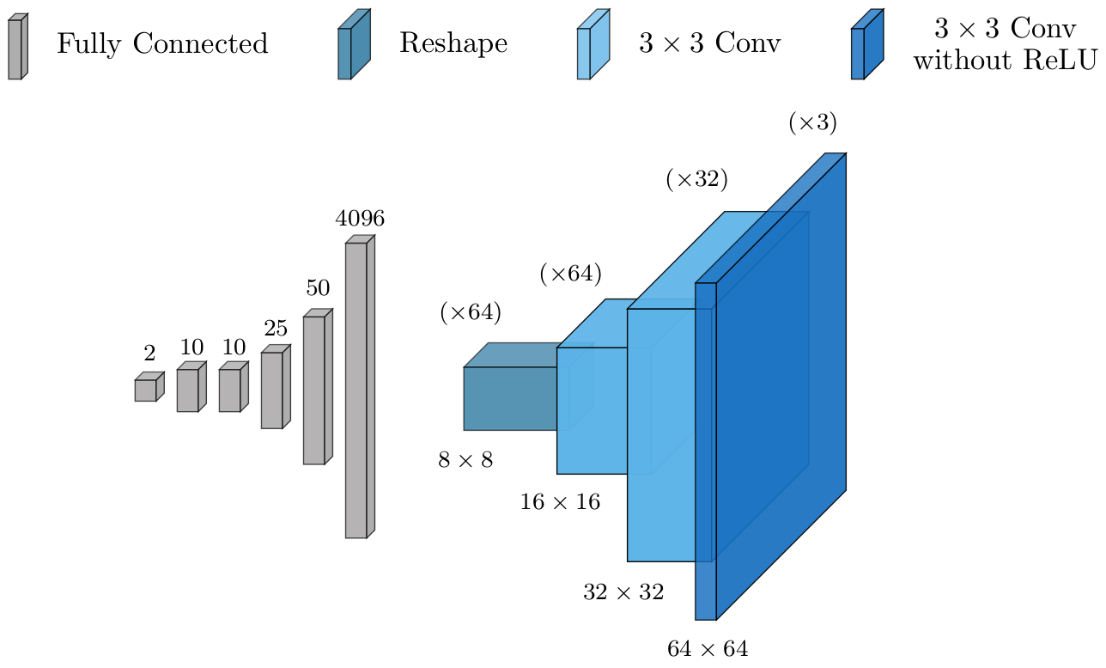


Fig. 6.4. Architecture of neural network for the forward problem.

To accomplish the task of predicting damage patterns from known hitting location coordinates, we propose the use of a deep neural network which consists of several fully-connected network layers followed by a sequence of transpose convolutional layers. In this framework, the input coordinates are first processed by the initial dense layers to construct features which are reshaped into a collection of coarse 8 × 8 resolution features. These coarse features are then passed to the transpose convolutional layers to be upsampled into a single, higher resolution 64 × 64 damage pattern prediction. The precise network structure used is shown in Fig. 6.4. This network was implemented using the TensorFlow software library and written in Python.

Table 6.3. Average computation times, MSE, L^1 errors for the forward problem model.

Dataset	Avg. Computation Time	Avg. MSE	Avg. L^1 Error
Training	0.003973 s	0.010214	0.054590
Testing	0.003970 s	0.010298	0.054779

Of equal importance to the network structure is the choice of a suitable loss function. In order to successfully train the network for the forward problem, it was necessary to design a loss function which converted the raw LAMMPS damage data into a format which is compatible with the deep learning framework and array/tensor-based format of the TensorFlow network implementation. This conversion was performed most naturally by bucketizing the atoms into a uniform grid of resolution 64 x 64. The average damage of the atoms contained in each bucket/bin of the grid was calculated and used to create the target outputs for the network. It was also necessary to account for atoms which had been substantially dislocated by a direct impact with the indenter and had left some bins at the disk's interior empty. These interior bins were assigned a full damage value of 1.0 to indicate that the atom which had originally occupied the space was displaced by the indenter. This was achieved by storing a copy of the LAMMPS data before the indenter strikes to serve as a template which indicates where the atom damages should be calculated. The mean square error (MSE) was then calculated on the interior of the disk using the template file as an indicator of where the interior error should be calculated:

$$Loss(y^*, y) = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} \mathbf{1}_T[i,j] * |y^*[i,j] - y[i,j]|^2 \qquad (33)$$

Here we denote the network prediction by y^*, the true damage pattern by y, the selected output resolution by N, and the Boolean template file by $\mathbf{1}_T$ (i.e. $\mathbf{1}_T[i,j]=1$ when the (i,j)th bin was originally occupied by an atom, and $\mathbf{1}_T[i,j]=0$ otherwise). All network layers used ReLU activation functions except for the final layer; as the damage pattern values are scaled to be within the range between 0 and 1, the final network layer was equipped with a sigmoidal activation function:

$$sigmoid(x) = \frac{1}{1 + \exp{(-x)}} \tag{34}$$

The model was trained using the standard ADAM optimization algorithm and backpropagation as implemented in TensorFlow. The learning rate for the ADAM optimizer was initialized to 0.0001 with a geometric decay rate of 0.9 applied every 3 epochs. The model was trained for 100 epochs using 20,200 training data points, with the remaining 5,050 data points reserved for testing/validation.

6.2.2 Results and Discussion for the Forward Problem

After training, the neural network's damage predictions are seen to closely approximate the true damage patterns from the LAMMPS simulations. The offline training procedure required just under 7 hours to complete using 4 CPUs; once completed, network predictions for damage patterns can be computed in less than 0.0025 seconds with average MSE and L^1errors in the range of 0.01 and 0.05, respectively (see Table. 6.3). Of note is the fact that a single LAMMPS simulation for the specified problem setup required an average of 6 seconds to complete on the same machine; the trained network is thus seen to provide approximations to the true simulation results while reducing the computation time by a factor of over 1500.

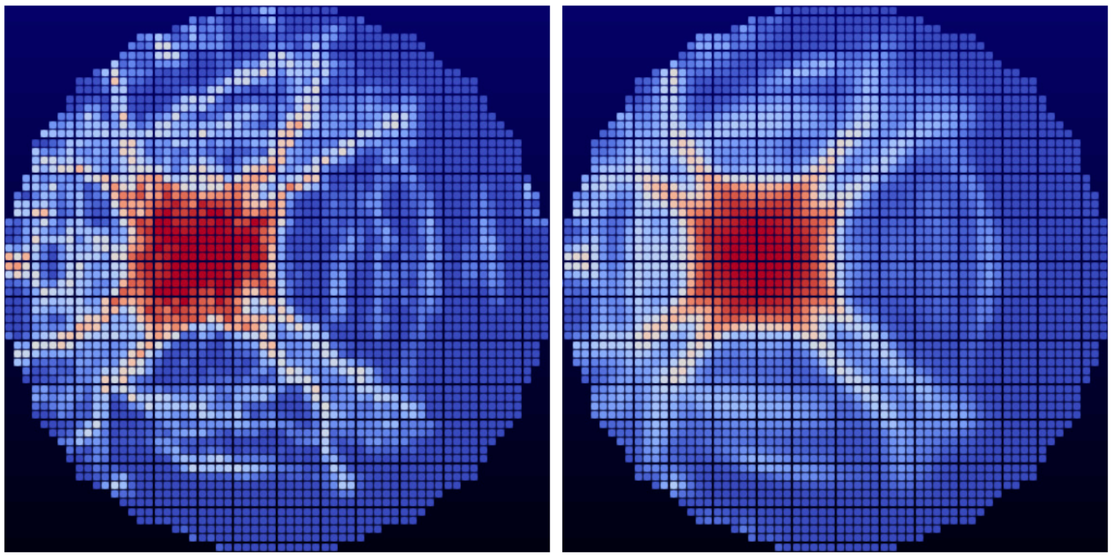


Fig. 6.5. Example of a true damage pattern computed using LAMMPS (left) along with the corresponding neural network prediction (right).

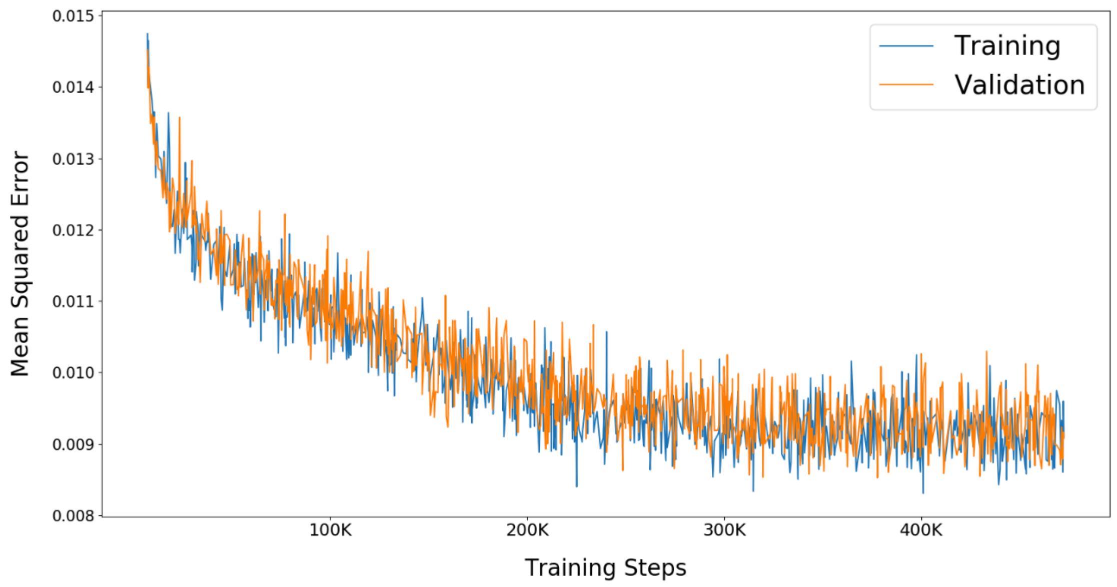
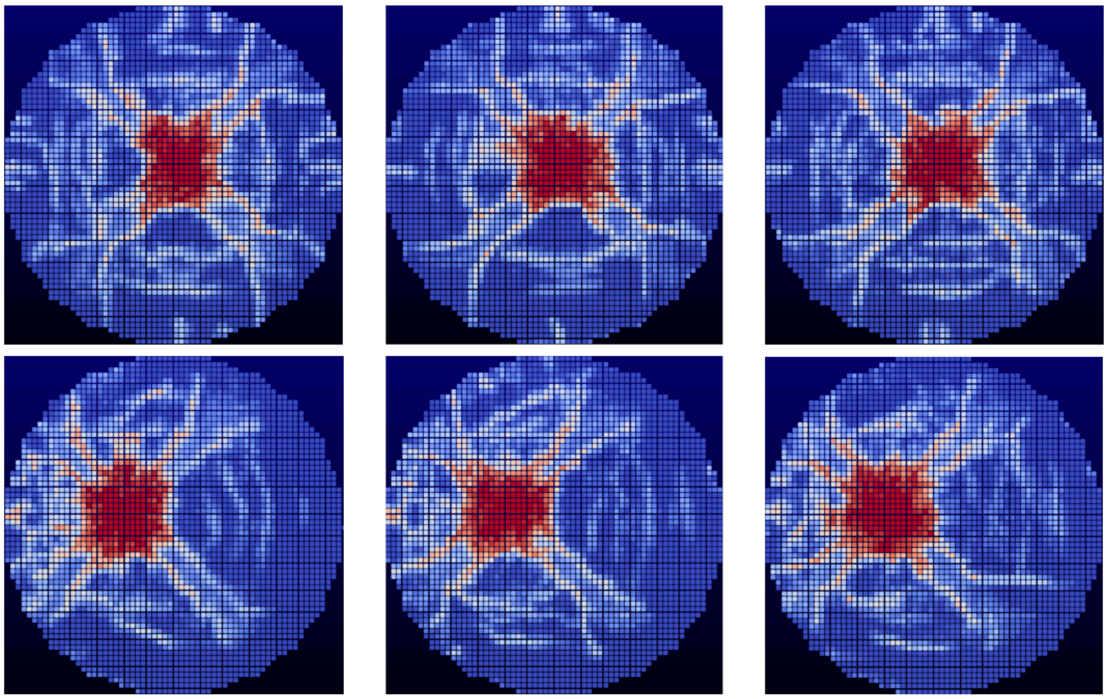


Fig. 6.6. Plot of the mean squared error of the network's predictions for training (blue) and validation (orange) examples throughout the training process for the forward problem.



Top-Left:	(0.001471, 0.002675)	Bottom-Left:	(-0.010236, 0.002842)
Top-Center:	(0.001603, 0.002598)	Bottom-Center:	(-0.010193, 0.002989)
Top-Right:	(0.001732, 0.002514)	Bottom-Right:	(-0.010275, 0.002694)

Fig. 6.7. Demonstration of low forward stability of true damage patterns for LAMMPS simulations. A cluster of neighboring hit locations near the center of the disk (top row) as well as a cluster further from the center (bottom row) both produce a widely varied collection of simulated damage patterns.

As depicted in Fig. 6.5, we see that the overall pattern of the damage on the disk is predicted quite well, however many of the finer details have been deemphasized or smoothed out by the network. The network predictions also tend to be more symmetric than the simulated patterns and tend to produce more linear cracking branches in contrast to the more jagged cracks from the simulations. The reason why the predictions are overly symmetric is most likely due to the relatively low forward stability of the LAMMPS peridynamics simulations. That is to say that the damage patterns are observed to change significantly even when the hit location of the indenter is changed by a very small amount. For example, the patterns shown in Fig. 6.7 correspond to hit locations which are very close to one another near the center of the disk along with a cluster of hit locations further from the center.

Each of these clusters has a maximum distance of less than 0.00031mm on a disk of radius 0.037mm (i.e. less than 0.85% of the radius), yet the simulated damage patterns are seen to be quite disparate. It is suspected that the smoother, more symmetric neural network predictions illustrated in Fig. 6.5 may be reflective of the average damage incurred when the indenter's hit location is varied over a small neighborhood of the precise location provided to the network as input.

CHAPTER 7. THE ANALYSIS OF THE CRACK PATTERNS USING THE NEURAL PROCESSES

7.1 Introduction

A significant problem to solve in solid mechanics is associated primarily to the construction of discontinuities in areas that do not exist at first. A typical example of the formation of discontinuities is a mechanical component failure due to cracking and breakage due to the dynamic load, high-temperature gradient, shock, explosion, etc. Occasionally, due to human negligence and mistakes in daily life, possessions of people are damaged causing a breakdown. In addition, automobile accidents can cause cracks in the car body. Finally, In DOE (Department of Energy), the blade of thermoelectric power plant and wind power plant which are formed the crack due to the influence of high temperature or wind, and reactor of the nuclear power plant are damaged by nuclear fission. Research on the cause of damage to objects with motility is actively underway. In general, many studies on fracture are performed by Finite Element Method (FEM).

However, FEM has a fatal error. First, the FEM is calculated based on partial differential equations, but, there are no partial derivatives on the crack surface. Also, only the approximate solution can be obtained in the calculation of the FEM, the result depends on the mesh, and the user's mistake is fatal to the result. In order to secure these defects, we used the peridynamics. The peridynamics theory was first introduced by S. Silling. By applying integral equations directly to the surface of the crack, the peridynamics can solve the absence of partial derivatives and compensate for the fatal error of the FEM. From this point of view, the peridynamics is the most appropriate theory for studying cracks or damage. The peridynamics theory is applied not only to cracks but also to various materials in which cracks are generated.

For instance, membranes and nanofiber, composites and brittle materials and prediction of viscoelastic materials. Based on the peridynamics theory, the MD simulation tools of LAMMPS and Peridigm [77] have been used extensively in crack research and have been actively researched to apply various material properties to crack simulations. For instance, PMB, LPS [78], and VES [79] models are set up in the MD simulation tool. Besides, we applied the deep machine learning method to reduce the computational cost to compensate for the remarkable increasing the computational times of the MD simulation as the size of the simulation increases. In recent years, Convolutional Neural Network (CNN) has been shown to provide superior light-weight to replace existing fully connected networks, particularly for highly structured data applications such as the peridynamics model. These networks have the advantage of being able to pre-train, learn the data sets generated from highly accurate models based on the peridynamics theory, and encode the process of the model into a neural network. This approximate model can be used to produce immediate results through the training and testing process. Also, we applied neural processes (NPs) [80] which combine the advantages of a neural network with the advantages of a Gaussian process with the data obtained from the peridynamics theory. Therefore, we can learn how to apply advanced information to the data, improve the efficiency of computation during the training and evaluation resulting in fast and more accurate results. Finally, in this study, CNN and neural processes are applied based on the data obtained from the peridynamics theory. As a result, we are trying to develop the peridynamics theory even more precisely with the case study for the characteristic of materials, and by applying the deep machine learning, we can speed up the calculation speed using the trained data and finally reduce the computational cost as a result.

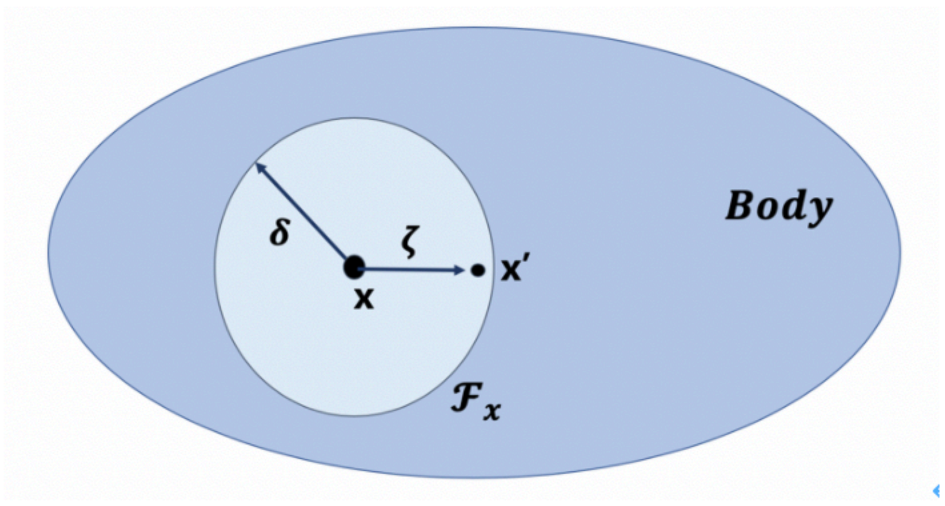


Fig. 7.1. Diagram of inverse problem and forward problem using machine learning.

7.2 The Peridynamics Model (PMB, LPS, VES)

In peridynamics, all points in the material are connected by bonds. The Fig. 7.1 shows the composition of the peridynamics model. Suppose that there is an x in the whole body. When we denote the circle with radius is δ (horizon) centered on x, all the points in it are connected to each other, and this is called the family of x. Therefore, peridynamics is the continuation of the classical continuum mechanics. The most crucial point of peridynamics is that it uses the integral equation.

Compared to the FEM study, FEM uses the partial differential equation for crack analysis. However, there are no partial derivatives on the crack surface. Therefore, peridynamics can solve the absence of partial derivatives by applying integral equations directly to the surface of the crack. From this point of view, peridynamics is the most appropriate theory for studying cracks or damage. The following is the equation of motion based on the peridynamics model. The results are as follows.'

$$\rho(x)\ddot{u}(x,t) = \int_{\mathcal{F}_x} \omega(u(x',t) - u(x,t), x' - x) dV_{x'} + b(x,t), \qquad t \geq 0 \tag{35}$$

The left term represents the equation of motion, ρ represents the mass density and $\ddot{u}$ represents the acceleration. In the right term, $\mathcal{F}_x$ is the family of x, T is the pairwise force function of the bond which connects point x and x'. u is the displacement of the point x according to the time domain. Finally, b represents the external body force density. The following definitions of ζ and η are needed to define conservation of linear momentum and conservation of angular momentum. ζ is the position vector in the initial arrangement and η is the relative displacement vector.

$$\zeta = x' - x \tag{36}$$

$$\eta = u' - u \tag{37}$$

$\zeta + \eta$ represents the current relative position vector between the points. The T function follows the two properties:

$$T(-\eta, -\xi) = -T(\eta, \xi), \qquad (\xi + \eta) \times T(\eta, \xi) = 0 \ \ \forall \eta, \xi \tag{38}$$

Eq. (38) represent the conservation of linear momentum and the conservation of angular momentum respectively. Conditions of T satisfy the conservation of linear momentum and the conservation of angular momentum.

In this study, the crack patterns were formed by applying the above-mentioned peridynamics theory. PMB, LPS, and VES models are applied to do the case study for crack patterns according to the material types. First, the PMB model is a Prototype Microelastic Brittle model which evolved from a model of isotropic and microelastic. The strength of bond depends only on bond stretch s. The equation is as follows:

$$y = |\zeta - \eta| \tag{39}$$

$$s = \frac{|\xi + \eta| - |\xi|}{|\xi|} = \frac{y - |\xi|}{|\xi|} \tag{40}$$

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 definition of PMB model is as follows.

$$F(y(t), \xi) = g\big(s(t, \xi)\big)\mu(t, \xi) \tag{41}$$

$$g(s) = cs \quad \forall s \tag{42}$$

$$\mu(t, \xi) = \begin{cases} 1 & if\ s(t', \xi) < s_0 \quad for\ all\ \ 0 \le t' \le t \\ 0 & otherwise \end{cases} \tag{43}$$

The function F is a product of g and μ functions, and the g function is a linear scalar-valued function consisting of c and s, which are spring constants and bond stretch, respectively. The μ function, a component of the f function, as a function of time and position vector, exists the conditions. If the bond stretch is less than a critical bond stretch s_0, u is 1, and otherwise, it is defined as 0. c, and s_0 are as follows.

$$c = \frac{18K}{\pi\delta^4} \tag{44}$$

$$s_0 = \sqrt{\frac{10G_0}{\pi c\delta^5}} = \sqrt{\frac{5G_0}{9K\delta}}\ , \quad G_0 = \frac{\pi c s_0^2 \delta^5}{10} \tag{45}$$

The spring constant c consists of bulk modulus K and δ(horizon boundary in peridynamics model), critical bond stretch s_0 is a function of shear modulus G_0 and horizon δ. Finally, shear modulus G_0 consists of spring constant c, and critical bond stretch s_0 and horizon δ in the boundary. The scalar force state of the PMB model is as follows:

$$\underline{t}_{PMB} = \frac{1}{2}\frac{18K}{\pi\delta^4}\frac{||\eta + \zeta|| - ||\xi||}{||\xi||} \tag{46}$$

Finally, briefly describing the LPS and VES models, the elastic properties of both models are defined by the bulk modulus and the shear modulus along with the horizon boundary. However, the VES model requires additional relaxation parameters and time constants. Based on the above description, the scalar force state of the LPS model and the VES model can be described as follows. For LPS model,

$$\underline{t}_{LPS} = -\frac{3K\theta}{m}\underline{\omega x} + \alpha\underline{\omega e^d} \tag{47}$$

m, θ, e and e^d are weighted volume, dilatation, extension state and deviatoric extension state respectively. The bulk modulus is K and the shear modulus G related term $\alpha = \frac{15G}{m}$.

$$\underline{t}_{VES} = -\frac{3K\theta}{m}\underline{\omega x} + (\alpha_\infty + \alpha_i)\underline{e}^d - \alpha_i\underline{\omega e}^{db(i)} \tag{48}$$

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

7.3 The Neural Processes

This section introduces the neural processes. Neural networks are the nonlinear functions that are very easy to train and the Gaussian processes [81] provide the stochastic framework for learning the distribution of the wide range of nonlinear functions. Thus, in limited data, Gaussian processes can capture probabilistic nature and uncertainty. However, the neural networks are computationally much more scalable than the Gaussian processes so that they can handle the vast amounts of data. Therefore, the neural processes is the model that combine the advantages of the neural networks and the Gaussian processes and can complement each other's disadvantages. Fig. 7.2 represents the diagram of neural processes. The process of neural processes is examined through a diagram. The massive flow of neural processes is as follows. Information flows from the context points through the potential space z to new predictions with target points. z is a random variable makes neural processes a probabilistic model and captures uncertainty about the function. Once the model has trained, the posterior distribution of z can be used before making predictions at test time.

In summary, given data is largely divided into context points and target points. It predicts y_T^* using the given context and target points. To learn more about neural processes, first, context points pass through the neural networks h to produce their respective representations r_1, r_2, ... r_c. Next, all the r's from each context points are aggregated to obtain a single r. This r is used to parameterize the distribution of z.

$$p(z|z_{1:c}, y_{1:c}) = \mathcal{N}(\mu_z(r), \sigma_z^2(r)) \tag{49}$$

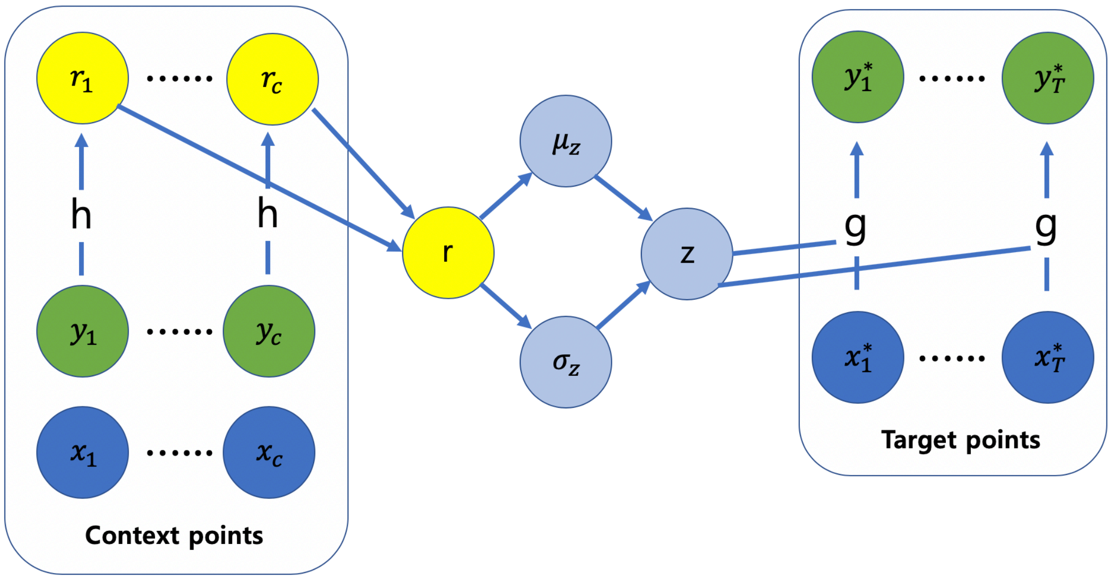


Fig. 7.2. The diagram of the neural processes.

Finally, to obtain the prediction point y_T^*, z is sampled, and the result is concatenated with the target point x_T^* to obtain a predictive distribution of y_T^* by passing through a neural network decoder (g). Inference of neural processes is performed in the variational inference framework. Inferring the approximate the posterior q(z|context, target), we can evaluate the prediction p(y_T^*|z, x_T^*). The approximate posterior q(z|context, target) is chosen for prediction y_T^* to use the neural networks h (encoder) and by using the same h, mapped the context points and the target points to obtain the aggregated r which in turn is mapped to μ_z and σ_z. Lastly, the parameters of the encoder (h) and decoder (g) are trained by the ELBO.

$$\text{ELBO} = E_{q(z|context,tatget)}\left[\sum_{t=1}^{T} \log p(y_t^*|z, x_t^*) + log\frac{q(z|context)}{q(z|context,\ target)}\right] \tag{50}$$

The first term in brackets is the expected log-likelihood of the target set. It is evaluated by the first sampling z ~ q(z|context, target). The second term of the ELBO has a normalization effect, which is a negative KL divergence between q (z|context) and q(z|context, target). This term indicates a summary of the contexts to be not too far from the summary of the targets.

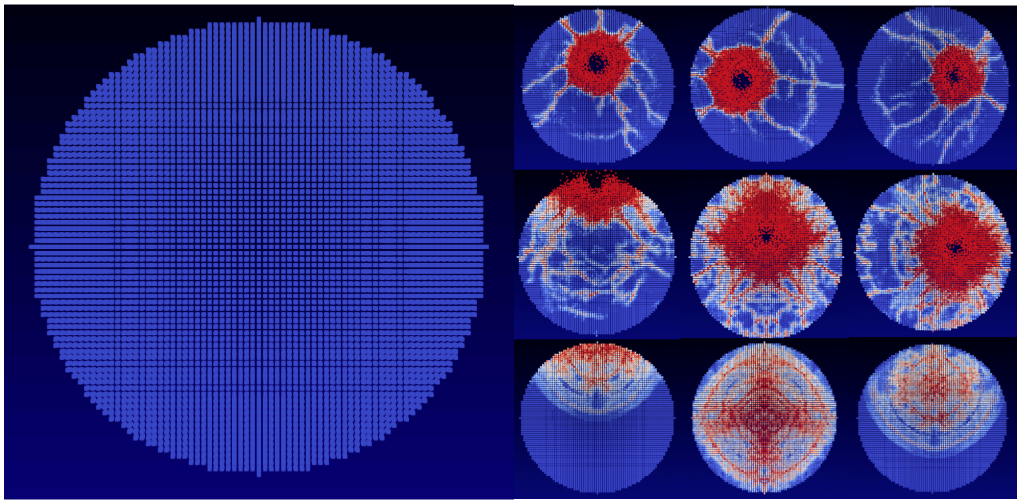


Fig. 7.3. Left side: Initial disk, Right side: Crack patterns (LPS, PMB, VES).

7.4 The Preparation of the Data Using the Peridynamics Theory and Models

In this study, peridynamics was applied to obtain the crack patterns of a moving disk and LAMMPS was used for the simulations. PMB, VES, and LPS models were used for the case study of the peridynamics model in order to apply deep machine learning. The crack patterns were penetrated by the spherical indenter perpendicularly to the disk moving in the x, y, and z directions. In this study, data were obtained by changing 4 parameters.

1) Hitting points are set by increasing and decreasing the x and y coordinates around the center of the disk, and the disk was penetrated by the indenter evenly throughout the cylindrical disk.
2) The parameters of velocity were changed to 100m/s and 100.1m/s with a slight difference. The reason for this was to find out how to classify the velocity of subtle differences when classifying the modes by applying deep machine learning.
3) The radius of indenter was adjusted by 0.007m and 0.008m.
4) To obtain the crack pattern of the moving disk, not the static disk, we changed the moving direction of disk (x, y, z).

Finally, the cylindrical disk was composed of 103,110 particles with a radius of 0.037 m and a thickness of 0.0025 m, each particle have a volume fraction of V_i = 1.25 x $10^{-10}m^3$ and the density of the disk material is ρ = 2200 kg/m^3. So far, 3 changed parameters have been applied to the moving disk, and the disk was composed of PMB model, VES model and LPS model for the case study. First, to apply the PMB model, `peri / pmb' pair style was used. The pair constants for the simulations have been specified as follows: c = 1.6863 × 10^{22}, horizon δ equal to 0.0015001, s_{00}= 0.0005, and α = 0.25. c is the spring constant for peridynamic bonds, the horizon δ is a cutoff distance for non-local interactions, the constants s_{00} and α correspond to material-dependent parameters for the critical bond stretch. Second, we used `peri / lps' pair style for the LPS model which consists of bulk modulus (K), shear modulus (G), horizon, s_{00} and α. Bulk modulus (14.9 × 10^9), shear modulus (14.9 × 10^9), horizon δ (0.0015001), s_{00} (0.0005) and α (0.25) were used to generate the crack patterns of LPS model. Finally, to set up the VES model, `peri / ves' pair style which consists of bulk modulus (K), shear modulus (G), horizon, s_{00}, α, m-lambdai and m-taubi was used. m-lambdai and m-taubi represent the viscoelastic relaxation parameter and time constant, respectively. Lastly, we generated a crack pattern for VES model by applying the bulk modulus = 14.9 × 10^9, shear modulus = 14.9 × 10^9, horizon = 0.0015001, s_{00} = 0.0005, α = 0.25, m-lambdai = 0.5, and m-taubi = 0.001. By this time, I have created data set for deep machine learning, and have studied the models and simulation settings for the case study research. We calculated the simulations for 1000 time-steps in one simulation and stored the output as a dump file, assuming that 1000 time-steps were meaningful by turning the simulation one by one for the case model to obtain the more sophisticated crack patterns. Finally, the pizza.py toolkit was used to change the EnSight data format to visualize the crack patterns using the dump files, then apply it to the Paraview [82,83] tool to visualize the crack patterns. In Fig. 7.3, the figure on the left is the initial model of the disk before the crack patterns were formed, the first line, the middle line, and the last line are the crack patterns were generated using the PMB, LPS VES models respectively.

7.5 Method and Results

This study used the supervised machine learning with the convolutional neural networks and the neural processes to classify the crack patterns in cases according to PMB, LPS, and VES models and 2-D regression problem. We have applied the deep machine learning method to improve the MD simulation using the peridynamics more efficiently, and we also investigate how accurately the peridynamics model can be predicted through the machine learning method. In this section, we will learn how to define the data set to perform supervised machine learning, how to label the train, validation, test data sets, how to set up the structure in the CNNs and neural processes. Finally, after training, validation, and testing, the loss function is calculated to determine how well the training and validation tests are performed. We will proceed with case studies through the success rate and the result of how accurately we predicted.

7.5.1 The Preparation of the Data for Classification

Table 7.1. 12 modes of the input image data for classification.

Mode	4 variables
	Radius of indenter (r) = 0.007, 0.008m Velocity of indenter (v) = 100, 100.1m/s Hitting location of disk (x, y) Moving direction of the disk (x, y, z)
Mode 1	r = 0.007m, v = 100m/s, Moving direction (x)
Mode 2	r = 0.007m, v = 100.1m/s, Moving direction (x)
Mode 3	r = 0.007m, v = 100m/s, Moving direction (y)
Mode 4	r = 0.007m, v = 100.1m/s, Moving direction (y)
Mode 5	r = 0.007m, v = 100m/s, Moving direction (z)
Mode 6	r = 0.007m, v = 100.1m/s, Moving direction (z)
Mode 7	r = 0.008m, v = 100m/s, Moving direction (x)
Mode 8	r = 0.008m, v = 100.1m/s, Moving direction (x)
Mode 9	r = 0.008m, v = 100m/s, Moving direction (y)
Mode 10	r = 0.008m, v = 100.1m/s, Moving direction (y)
Mode 11	r = 0.008m, v = 100m/s, Moving direction (z)
Mode 12	r = 0.008m, v = 100.1m/s, Moving direction (z)

For the convolutional neural networks, Matconvnet, a MATLAB toolbox was used. First, the crack pattern was stored as the dump files through the MD simulation based on the peridynamics theory. These dump files were converted to the 64×64 Numpy array and saved them as the data. There are three types of data: training data set, validation data set, and testing data set. The total number of dump files is 10800, and 12 modes of data were obtained by changing the 4 parameters mentioned in section 4. 12 modes are represented in Table 7.1. We also randomly shuffled all data to reduce the imbalance of the data and also correctly divided the number of data according to each mode and assign it to do training, validation, and test data set. Also, 30% of the training data was designated as a validation data set to observe the overfitting and evaluate the training. Through these processes, we have used 6300 training data sets, 2700 validation data sets, and 1800 test data sets. In neural processes, Pytorch [84] which is a deep-running implementation library for training and testing was used and crack dataset was made the same as the MNIST [85] data format. Through the MD simulation using the peridynamics theory, 10800 dump files were obtained as output. These are converted to the Numpy array and converted them to 28×28 png image files. Finally, 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 the label similar to MNIST dataset. Through these processes, 9000 training data, 9000 training labels, 1080 test data and 1080 test labels are obtained.

7.5.2 The Composition of Data Set for CNNs and the Neural Processes

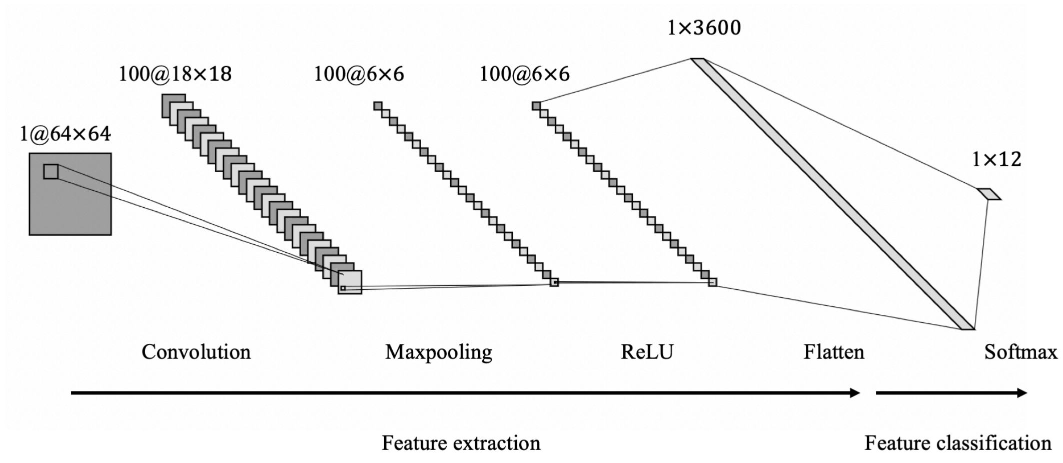


Fig. 7.4. Architecture of CNNs for classification of modes.

For convolutional neural networks, the training data set and the test data set were made into a .mat file to apply the data to Matconvnet. Each set also contains a 30% validation set. The .mat file consists of `images' and `meta' format. First of all, `images' has a structure of `labels', `set', `data'. `labels' are used to display the labels for randomly mixed data. In `labels', our datasets are labeled randomly from 1 to 12 to classify as 12modes. `Set' specifies the training data set, test data set, and validation data set. For example, 1, 2, 3 can be referred to as training data, test data, and validation data respectively. `data' stores the image of the crack pattern with Numpy array in the form of a 4-D single. For instance, if the size of 4-D single is $64 \times 64 \times 1 \times 10800$, there are 10800 grey images with 64×64 pixels. The `meta' structure contains `set' and `classes'. `set' is related to `set' in `images', where 1 is training data, 2 is test data, and 3 is validation data respectively. Finally, `classes' specifies from 1 to 12 related to the mode of the data we want to sort. Also, our data was made similar to the MNIST data type to apply the neural processes. All data were compressed in ubyte.gz in the form of training sets, training labels, test sets, and test labels, respectively. Likewise, the data was randomly mixed before storing the data and labels. Since the validation data set does not exist, it is possible to specify the validation data set to use the portion of the entire training set. The labels exist from 1 to 12, which are the same as the number of the classification. The data set was arranged to be easy to see at a glance by attaching each images side by side with 28 × 28 pixels like MNIST data type. All of these data types are ndarrays, an N-dimensional array.

7.5.3 Convolutional Neural Networks for Classification

Deep learning has become a powerful tool for machine learning, and it is possible to perform various cognition and inference by appropriately utilizing the human knowledge existing in annotated data. CNNs are widely used to solve many computer vision problems that are image classification and object recognition. These neural networks are composed of several layers of neurons between the input and output, which can be shared with neurons in other layers, making it very easy to communicate information. Each connection unit acts as a linear or nonlinear operator defined by the set of parameters. We used the Alexnet structure to classify the modes of the crack patterns. There are two processes for classifying properly annotated data. The first is the feature extraction step. This step takes a gray 2D image of a × a pixels as an input. This input image forms a 1 × c feature vector through each layer of neurons. The second is the feature classification. This process is the process of classifying the feature vector into the desired modes. Through these two processes, images can be classified. Let's take a look at this process in more detail. First, the process of the feature extraction step requires three layers. The first is the convolutional layer. The convolutional layer receives the input image of $W_1 \times H_1 \times D_1$ and produces an output image of size $W_2 \times H_2 \times D_2$. Four hyperparameters are required in this process. Hyperparameters consist of the horizontal and vertical spatial size (F), stride (S), zero padding (P) and the number of the filter (K). The input image forms an output image by the relation of 4 hyperparameters, and the relation equations are as follows.

$$W_2 = (W_1F + 2P)/S + 1 \tag{51}$$

$$H_2 = (H_1F + 2P)/S + 1 \tag{52}$$

$$D_2 = K \tag{53}$$

The volume of the output image is determined through these relations. Let's describe our calculations through the Fig. 7.4. When an input image of 64 × 64 × 1 passes through the convolutional layer using the above-mentioned relation equations with hyperparameters of 11 × 11 filter size and 100 filters, stride (1) and zero padding (0), the 18 × 18 × 100 output images are obtained. The second is Max pooling layer. The Max pooling layer receives the input image passed through the convolutional layer and downsample it. Therefore, it reduces the computational load and memory usage and makes calculations faster. By applying our study, we set the Max pooing to 3 in this study. Thus, using the Max pooing layer to an image of 18 × 18 × 100 obtained through the convolutional layer, input images are converted to 6 × 6 × 100. Finally, it is the ReLU layer. The ReLU layer is an activation function that is applied to the neuron at each pixel location. Also, by changing all values less than 0 to 0, the computation speed is significantly increased. The ReLU activation function is defined as follows:

$$ReLU(x) = \max(0, x) \tag{54}$$

Where x is the intensity of a pixel. It controls how much information is passed through the network. So far, we have looked at the process of feature extraction. Finally, when we look at feature classification, the output image of 6 × 6 × 100 obtained through the three layers mentioned above was arranged in the form of 1 × 3600 vector through the flattening process. The reason is that our ultimate goal is to classify into 12 modes. Thus, a flattened 1 × 3600 vector is classified as 1 × 12 through the Softmax layer. SoftMax layer is often used in the final layer of a neural network- based classifier. It takes a vector of arbitrary real-valued scores (in z) and reduces it to a vector of values between 0 and 1.

This process is as follows:

$$z^{[L]} = \omega^{[L]} a^{[L-1]} + b^{[L]} \tag{55}$$

Compute z, we need softmax activation function

$$t = exp^{z^{[L]}} \tag{56}$$

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

$$a^{[L]} = \frac{expz^{[L]}}{\sum_{1}^{12} t_i} \tag{57}$$

Output(a) is going to be the vector t but normalized to sum to 1.

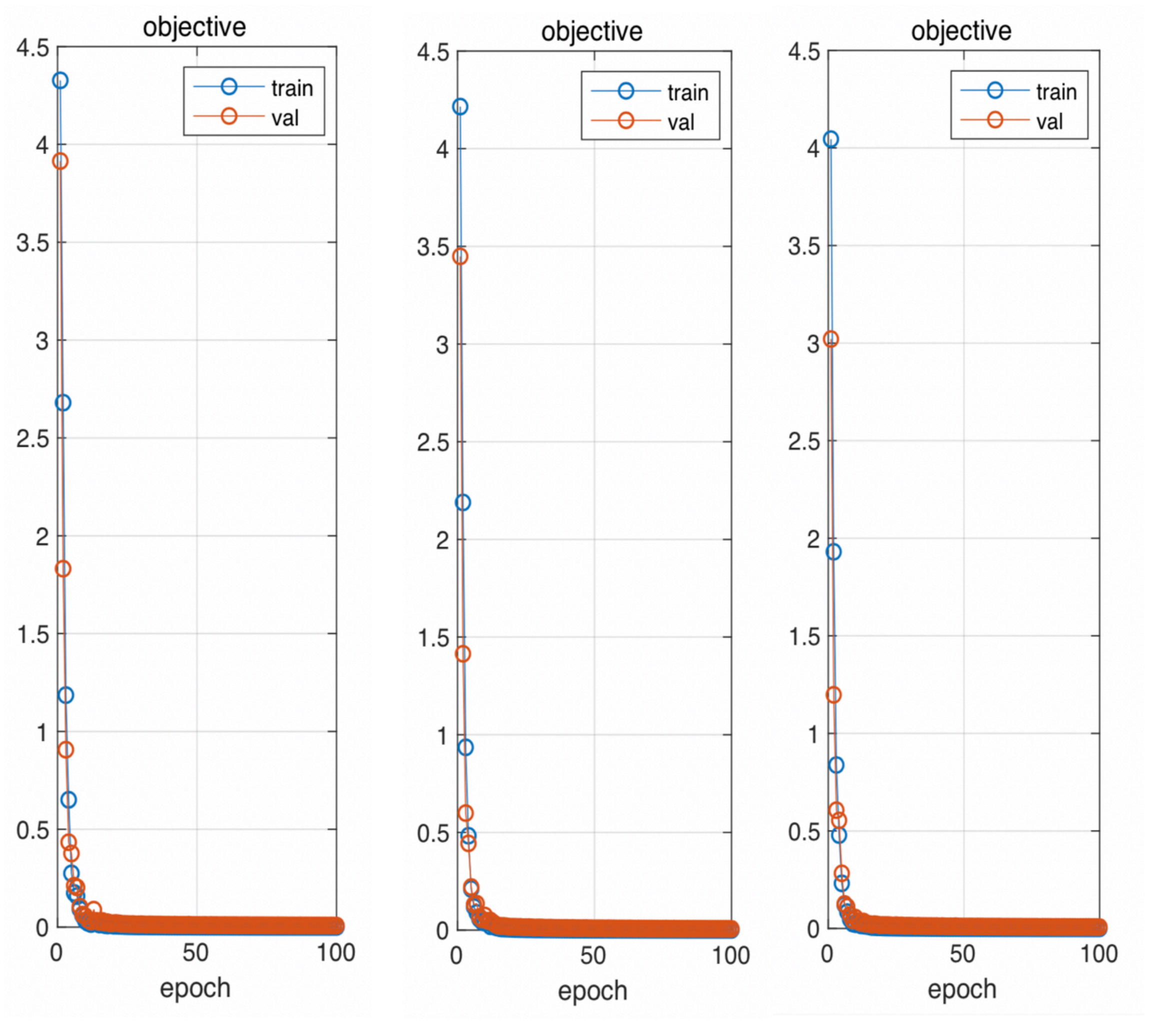


Fig. 7.5. The objective plots for PMB, LPS and VES model.

7.5.4 The Results of the Training and the Testing Using CNNs with Success Rates

In this section, we examined the results of training, validation, testing and success rate in classifying modes using specified data and CNNs. As mentioned earlier, in the 9000 training datasets, 30% of training datasets were used for validation datasets in the training process. Fig. 7.5 shows the training and validation results according to PMB, LPS and VES model. During the training, hyperparameters were set such as learning rate was 0.001, batch size was 50, and the number of the epoch was 100. As we can see in the Fig. 7.5, we observed the objective value and judged whether the training was proper or not. The blue line indicates a training error and the orange line shows a validation error.

The objective value is the same as the error graph, and there has some error when the training is first started. However, as the epoch increases, it gradually converges to zero. It is also possible to judge the overfitting of the training through the Fig. 7.5. If there has overfitting in the training process, the validation line may not converge to 0 or the value may drop and it may converge to a particular value. In this case, we can make the model more simple, or reduce the overfitting by regularization. In this study, as shown in the Fig. 7.5, training and validation errors converge to zero exactly without overfitting or underfitting. So far, we have studied the training and validation. Finally, we tested the 1200 images based on the training data and judged how accurately we predict the modes and classified the test image through the success rate.

Table 7.2. CNNs prediction of crack patterns using the Neural processes

The PMB test image number : Label (Mode)	CNNs prediction results (Mode)
Test image 1 : 4 (Mode)	4 (Mode)
Test image 2 : 11 (Mode)	11 (Mode)
Test image 3 : 5 (Mode)	5 (Mode)
Test image 4 : 3 (Mode)	3 (Mode)
Test image 5 : 7 (Mode)	7 (Mode)
Test image 6 : 2 (Mode)	2 (Mode)
Test image 7 : 4 (Mode)	4 (Mode)
Test image 8 : 8 (Mode)	8 (Mode)
Test image 9 : 1 (Mode)	1 (Mode)
Test image 10 : 2 (Mode)	2 (Mode)
The LPS test image number : Label (Mode)	CNNs prediction results (Mode)
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)
Test image 9 : 4 (Mode)	4 (Mode)
Test image 10 : 5 (Mode)	5 (Mode)
The VES test image number : Label (Mode)	CNNs prediction results (Mode)
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)
Test image 9 : 2 (Mode)	2 (Mode)
Test image 10 : 11 (Mode)	11 (Mode)

Table. 7.2 shows the classification results according to the model using the training data. The test image number and Label (mode) indicate the image data we tested and the labels associated with it. CNNs prediction results, on the other hand, show what the test images are predicted. In the results, the prediction of the test results according to the model was successful.

We obtained the success rate to numerically determine the test results and the relation equation is as follows:

$$\frac{number\ of\ the\ correct\ test\ images}{the\ total\ number\ of\ the\ test\ images} \times 100\% \tag{58}$$

The prediction results of the classification modes using the training data showed the success rates between 99.6% and 99.8%.

7.5.5 The 2-D Regression Problem Results Using the Neural Processes

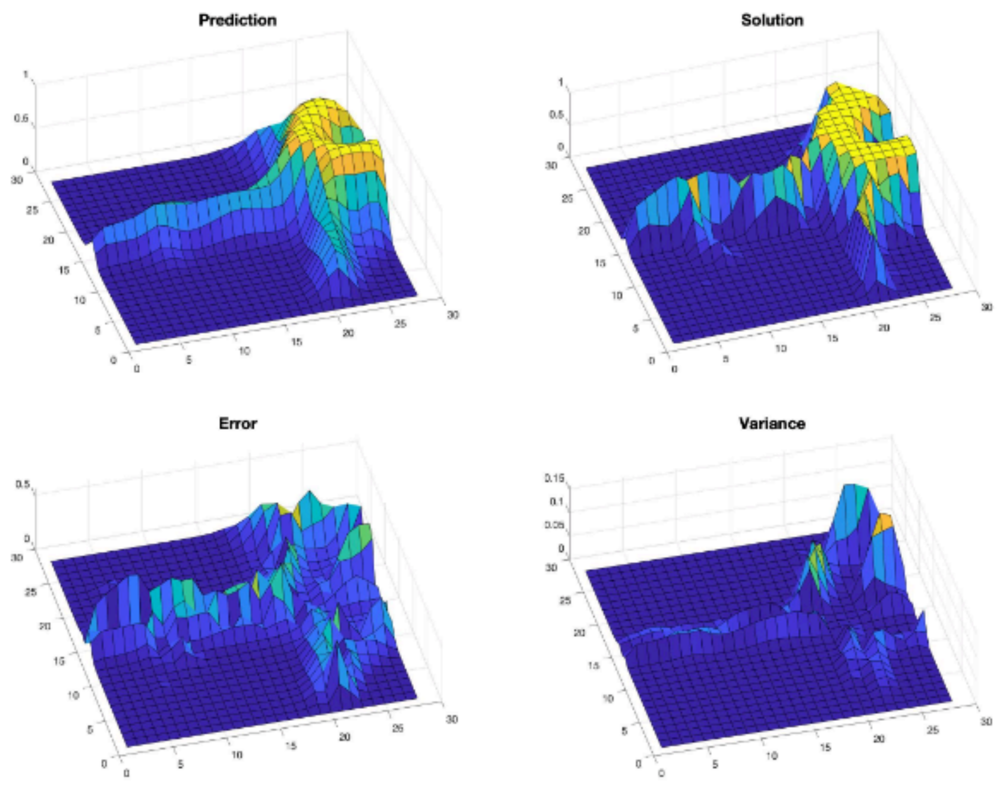


Fig. 7.6. These results evaluate the network's predictions for the crack patterns problem. This example has been taken from the test data set and have not been seen by the network during training. Prediction and true solution are shown at the top, with the corresponding absolute errors plotted below along with the network's predicted variance.

So far, we have classified the crack patterns according to the modes by CNNs. In this section, we try to predict the crack patterns by solving the 2-D regression problem using NPs. As mentioned earlier, the data used 9000 training data and 1800 test data. The structure of the data is similar to that of MNIST, and the size of each image is 28 × 28 pixels. The NPs were used to solve the 2-D regression problem and the experiment set up of the NPs was as follows. In hyperparameters, the input batch size for training was set to 128, and the number of the epoch was set to 3000.

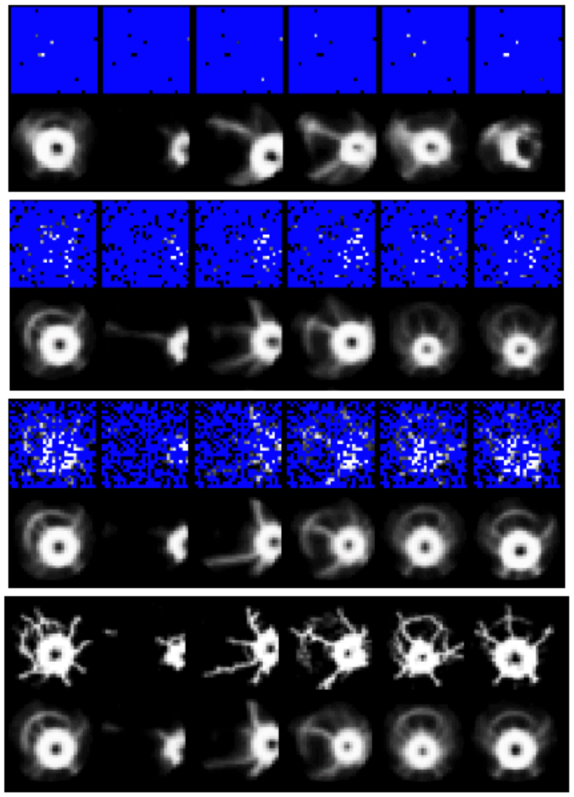


Fig. 7.7. 2-D regression problem results for crack patterns with 10, 100, 300, 784 contexts points. 1) First line represent the prediction results in epoch 1 compared to the context points (10, 100, 300, 784). 2) Second line represent the prediction results in epoch 500 compared to the context points (10, 100, 300, 784). 3) Third lines represent the prediction results in epoch 1000 compared to the context points (10, 100, 300, 784). 4) Last lines represent the prediction results in epoch 3000 compared to the context points (10, 100, 300, 784).

Also, the dimension of representation (r) was 128 to obtain each representation through the NNs from context points. This is associated with the linear layers of NNs which yield the 128 representations through three linear layers (3 × 400, 400 × 400 and 400 × 128). In NNs, we used the ReLU layer as an activation function and added efficiency to the training. Representation obtained through NNs is used to parametrize the distribution of z which is related to the Eq. 49, and through this process, the dimension of the global latent variable (z) is also 128. Using the obtained z and target points, we predicted the crack pattern through the g of the linear layers. The linear layers of the g are composed of 130 × 400, 400 × 400, 400 × 400, 400 × 1. As a result, crack patterns can be predicted.

Finally, the loss function is binary cross-entropy, and the relation equation is as follows:

$$H_{p(q)} = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot \log(p) + (1 - y_i) \cdot \log(1 - p) \qquad (59)$$

For instance, considering the modes, we assume that there are modes 1 and 2. y represents the label for mode 1 and p is the predicted probability of the point being mode 1 for all N points. This represents that for each mode 1 point, it adds log(p) to the loss, that is, the log probability of it being mode 1. On the contrary, it adds log(1-p), that is, the log probability of it being mode 2, for each mode 1 point. Finally, the model was trained using standard ADAM optimization algorithms and backpropagation implemented in Pytorch. The learning speed of the ADAM optimization was set to 0.001.

In this study, the network predictions were evaluated using the above network settings and the results are shown in Fig. 7.6. The network prediction and true solution can be found at the top of the line and can be seen to be quite similar when comparing the results. In next line, the absolute error can be confirmed by using the mean value of the prediction obtained from the distribution of z mentioned in Fig. 7.2 and true solution.

Finally, we showed the result of the variance obtained as the network progresses the prediction over the epoch 3000. As a result of the variance, the variance value was significantly lowered when the process of the prediction proceeded from epoch 1 to reach the epoch 3000, and the range was from 0 to 0.015. We can see the result that the prediction value becomes similar to the true solution value because the variance value becomes smaller according to the result of increasing epoch iteration and parameter tuning. The variance can be used as an indicator to show how well network's prediction is. Also, the location with the largest variance is the location where we should take additional samples. We can use it for experimental design to get additional optimal sample locations. Using the structure of the generated neural processes, the predicted results of one of our models, LPS model through the training and testing, are shown in Fig. 7.7. It shows the result of the prediction of the test data. In the training process, we set the context points to 10, 100, 300, and 784 and show the test results accordingly. The 1 and 2 lines of the figure show the result of 1 epoch according to the context points 10, 100, 300, and 784 from the left side. As you can see from the figure, the test images are not predicted yet since it is the first step in the training process. The 3 and 4 line are the results of anticipating the test images as the 500 epochs. In this process, the crack pattern is blurred, but the crack pattern can be predicted gradually. Finally, the 7 and 8 lines are the results of anticipating the test images after finishing the final training at epoch 3000. As we can be seen from the 7 and 8 lines, we accurately predicted the crack patterns and even more surprising is that also if the context points are 10, 100, and 300, the crack patterns are predicted precisely as well as the results with the context points of 784. Through this process, if NPs are used, even if there is not enough information, it is enough to predict the model.

7.6 Conclusions

We used the peridynamics to obtain the crack pattern changing the 4 variables: velocity of the indenter, the radius of the indenter, hitting location of the disk and moving direction of the disk. This data obtained the crack patterns from moving the disk and more detailed and accurate crack patterns using peridynamics theory which can compensate for the disadvantages of FEM. Besides, various models (PMB, LPS, VES) could be complemented by using case studies with convolutional neural networks using peridynamics. First, the modes of the crack pattern were classified, and the case study of models was conducted with convolutional neural networks. In this process, data obtained through the peridynamics have labeled the data according to each mode and did the supervised learning. By evenly distributing the data, the problem of the imbalance data was reduced. A training data set of 30% was designated as a validation data set to solve the most problematic overfitting in the training process. Overfitting was observed during the training and training was optimized by tuning hyperparameters to reduce the overfitting. As a result, 99.6% ~ 99.8% success rate was obtained, and the modes could be classified accurately.

Finally, the neural processes was used to solve the 2-D regression problem. The neural processes was optimized to address the regression problem by combining the advantages of neural networks and Gaussian processes. In this process, the data obtained through the peridynamics were made similar to the MNIST data and were classified into training data set and test data set for training. We also set the context points to 10, 100, 300, and 784 so that we could see the results of the test images. As a result, if the epoch is low, the prediction is not good enough regardless of context points. However, as the number of epochs increases, the result of the speculation is right.

In the final epoch 3000, even if the context points are 10, the prediction is sufficiently good as in the case of the context points 784. In other words, using neural processes, the forecast is possible using data that is not enough information.

CHAPTER 8. AUTOMATICALLY PARTICLE PICKING ON CRYO-EM IMAGING WITH DEEP MACHINE LEARNING

8.1 Introduction

Recent developments in Cryo-EM technology have had a great impact on structural biology and protein molecular complexes. However, high-resolution Cryo-EM studies of molecular complexes require the choice of a large number of high-quality particles. The particle selection stage in this process is very labor intensive. This is because it is manually selected by the human hand to obtain high quality particles. Also, Particle selection is also very subjective because it is done passively, and the results can be very different because particle selection is done by humans. Therefore, in this study, deep learning method was applied to reduce the effort of selection of high-quality particles. In recent years, deep learning technology has been applied to researches such as computer vision and natural language processing through the availability of large - scale learning data, the development of powerful computing platforms and learning algorithms. Therefore, this study proposes a learning frame structure using deep learning and aims at freeing passively selecting high quality particles as the ultimate goal.

8.2 Dataset Creation for Automatically Picking Particles in Cryo-EM Images

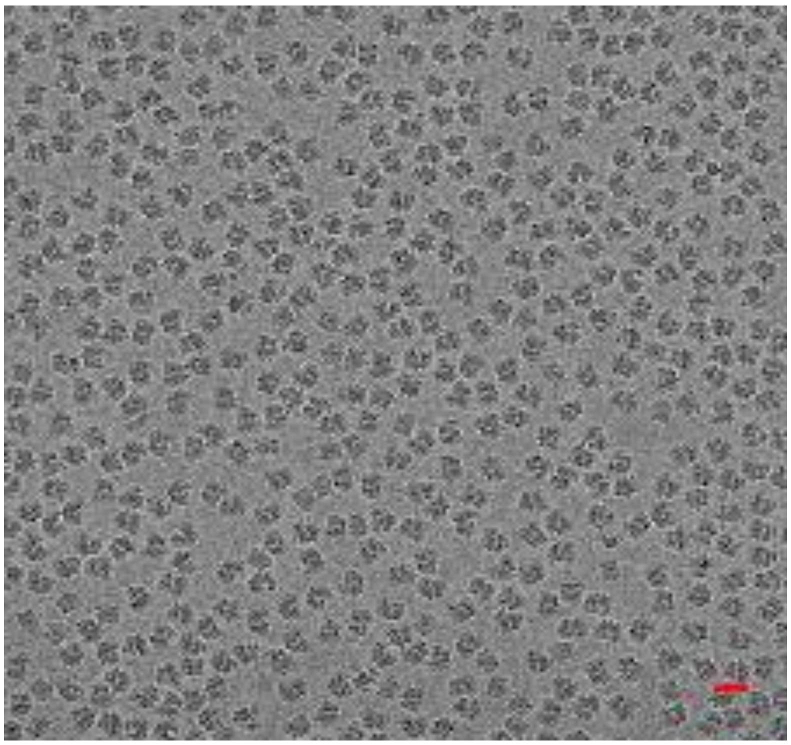


Fig. 8.1. Cryo-EM raw image from EMPIAR.

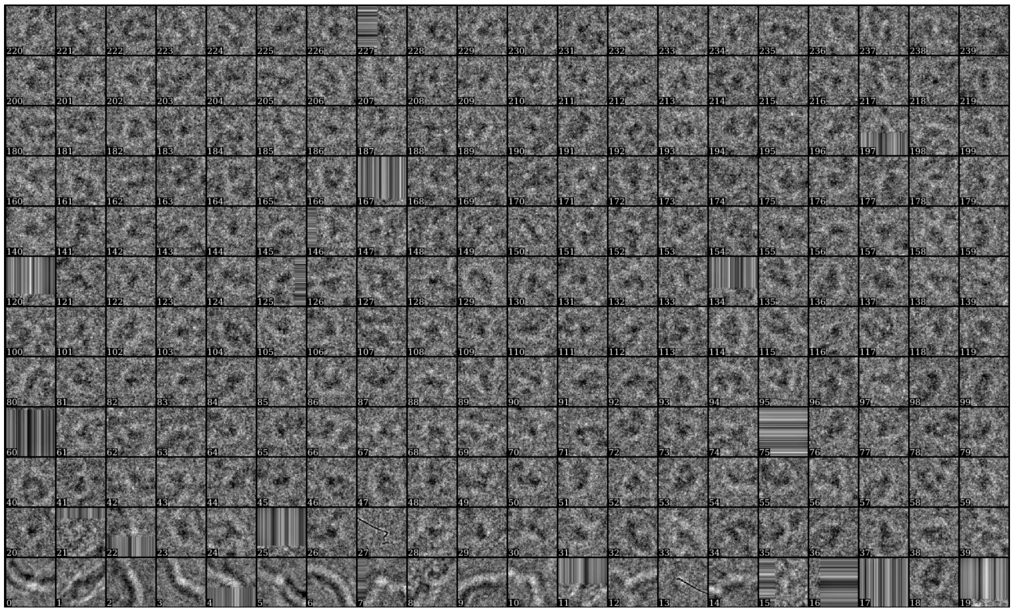


Fig. 8.2. Manually particle picking with EMAN2.

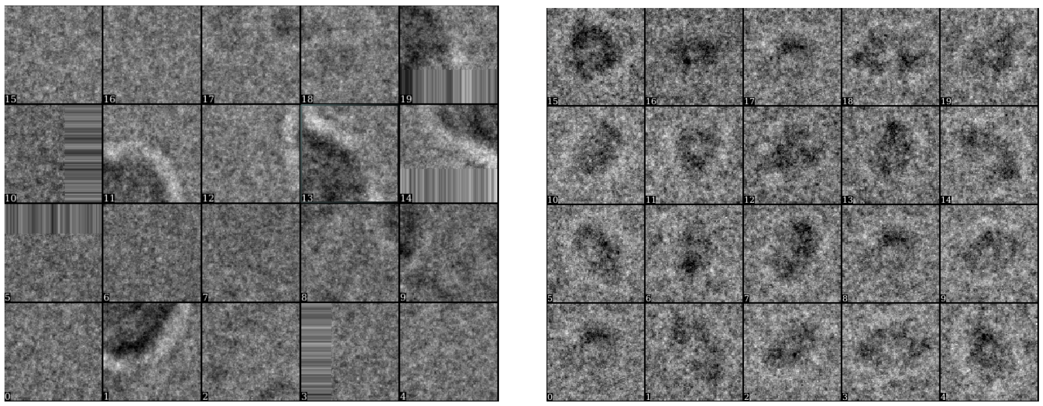


Fig. 8.3. Left side (Bad images), Right side (Good images).

In this section, we introduce the process of obtaining training data and test data to extract particles in the Cryo-EM image. The purpose of this study is to replace handwork to obtain a high-resolution image of the past using deep learning. Therefore, in order to utilize deep learning, training data, validation data, and test data are needed. We obtained a .MRC file via the EMPIAR [86] for the Cryo-EM image sample, and the image can be seen in Fig. 8.1. As you can see from the figure, we can identify hundreds of protein particles. We used the EMAN2 [87] tool to read the .MRC file from EMPIAR. Briefly, the EMAN2 tool reads the .MRC file and displays it as an image, and has an algorithm that automatically extracts particles from the tool itself. However, the accuracy was 60% when compared with our training data, and we can confirm different success rates depending on the image in EMAN2 so we only used this tool to read the data and obtain the data we wanted. The advantage of this tool is that we can extract and store the images which can control the pixel size we want. As you can see in Fig. 8.2, we can extract the data by 100 by 100 pixels and save it as deep learning data. However, this study has drawbacks in that it requires manual work to obtain the desired good and bad particles. Fig. 8.3 shows the results obtained by picking good and bad particles through the manual operation. On the left represents the bad particles which are the error part and the background part are specified in the image, and the right part is the result of extracting protein particles in the image with good particles. Through this work, we obtained 100 by 100 pixels 2600 images and collected various data by flip and rotate the data in order to increase the data. Therefore, 2000 training data and 600 test data were obtained, and 30% of the training data was designated as validation data. Finally, we used MATCONVNET which is based on the MATLAB for supervised learning. In order to do a training, we converted the image files into Numpy array and saved 4-D single data as .mat file. A detailed introduction to this data can be found in the next section.

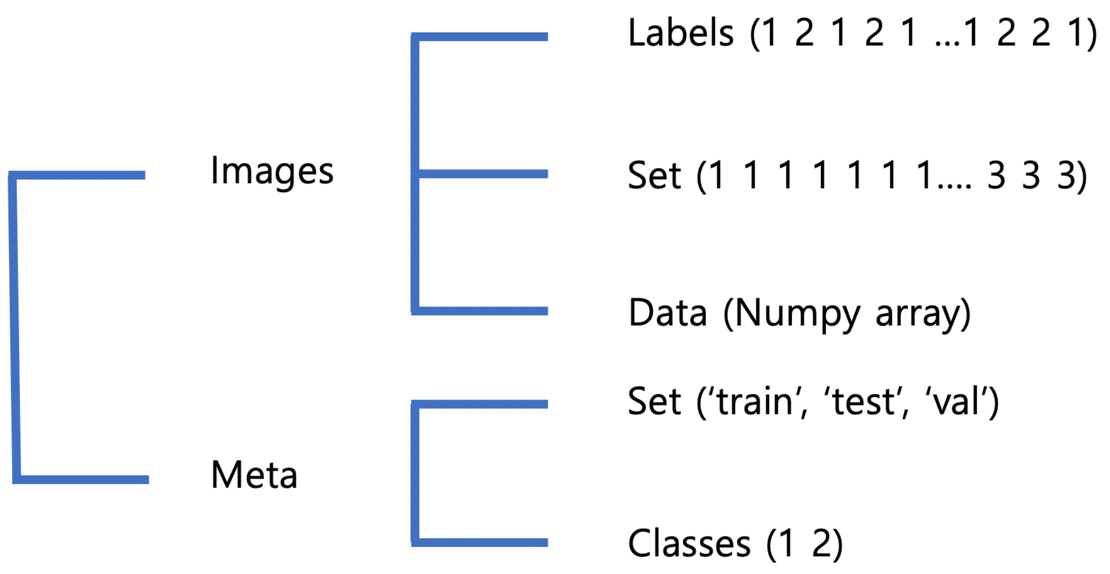

Images
Labels (1 2 1 2 1 ...1 2 2 1)
Set (1 1 1 1 1 1 1.... 3 3 3)
Data (Numpy array)
Meta
Set ('train', 'test', 'val')
Classes (1 2)

Fig. 8.4. Data structure in the .mat file.

8.3 Reference Annotation

The training data set and the test data set are made into a .mat file that can be applied to Matconvnet and its structure is shown in Fig. 8.4. The structure of the data in the .mat file is largely divided into `Images' and `Meta'. There are `Labels', `Set' and `Data' classes in the `Images' class. `Labels' displays labels on images of input data sets. Labels were displayed randomly from 1 to 2 since data has good and bad particles. In the set class, training data, test data, and validation data are indicated by 1, 2, and 3, respectively. In the `Data' class, the data obtained from the images of the good and bad particles are converted to a Numpy array and expresses it as a 4-D single matrix. Finally, there are `Set' and `Classes' in the `Meta' class. The `Set' contains the words `train', `test' and `val' which are related to the `Images' class and indicate that the 1 2 3 in the set of images class is training data, test data, and validation data respectively. The `Classes' in the `Meta' specifies a class from 1 and 2 due to the good and bad particles.

8.4 Convolutional Neural Networks for Particle Picking in Cryo-EM Image

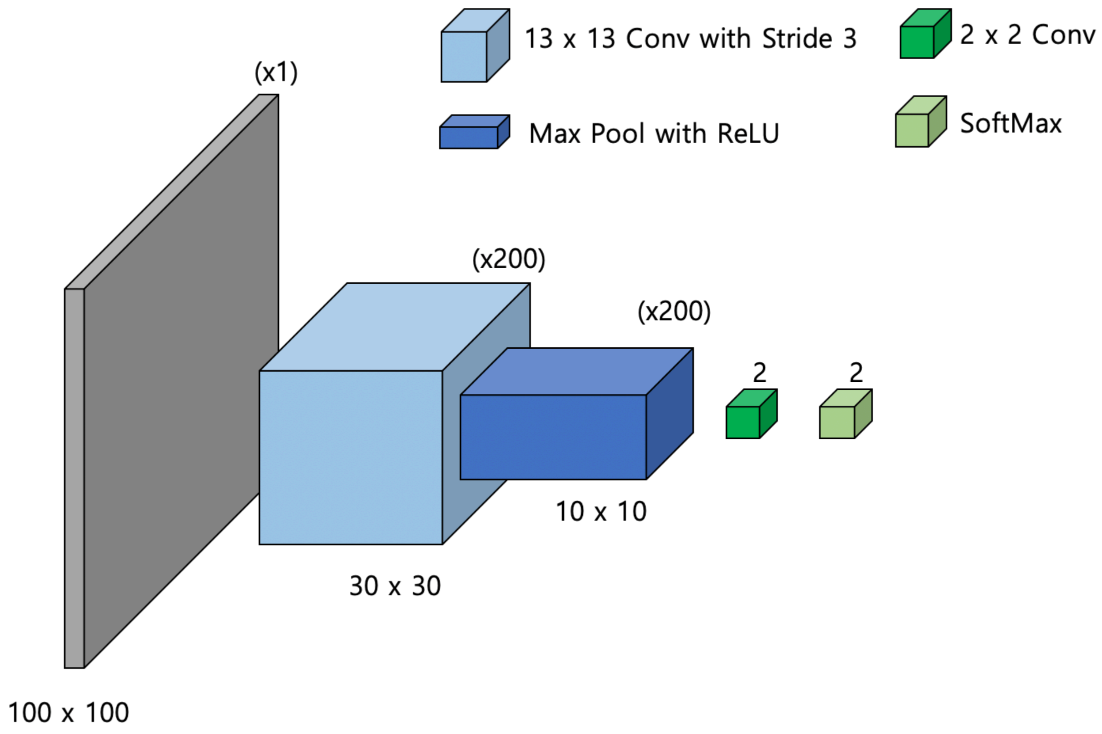


Fig. 8.5. Architecture of convolution neural network (CNN) for automatically particle picking.

Deep learning is a subset of AI and machine learning capabilities of achieving state-of-the-art results on problems such as object detection, text generation, and image recognition. Using existing human knowledge in the form of annotated data, various cognitive and inferential tasks can be performed. Convolutional neural networks are widely used to solve computer vision problems such as classification of images and object detection. These convolutional neural networks consist of several layers which can be tuned through the connection weights between input and output layers. In this work for damage analysis using machine learning, a similar structure of AlexNet are chosen. The CNN processes the input damage pattern in two steps: a feature extraction step carried out by the convolutional layers, followed by a classification step performed by the fully connected layers. First, feature extraction uses a 2D damage array of size $N_{row} \times N_{col}$ for input data and produces a vector of size $10 \times 10 \times 200$. Second, the feature vector is converted to a classification vector of size $1 \times N_c$, where N_c is the number of classification categories. This work accounts for two categories corresponding to the good and bad particles.

8.5 Feature Extraction and Classification

The purpose of image extraction and classification is to create a $1 \times f_c$ feature vector in relation to the input data, which is related to the 8 modes used for classification. The feature vector is generated from the input training data by applying successive filters and the filters are 1) max pooling layer, 2) ReLU layer and 3) SoftMax layer; each of the three types of network layers are described below.

1) Max pooling layer is a sample-based discretization process. The objective of max pooling layers is to downsample the input images in order to reduce the computational load and the memory usage. If the input image x is a square image of size $N_{row} \times N_{col}$, and a = 3, then the output image y of the pooling operation is of size $[N_{col}/a] \times [N_{row}/a]$. In our research, the downsampling rate is set to a = 3, the 100 pixels by 100 pixels array becomes 30 pixels by 30 pixels and then 10 pixels by 10 pixels.
2) The rectified linear unit (ReLU) layer applies an activation function which is defined as the positive part of its argument: where x is the input data to a neuron. A ReLU performs a threshold operation to each element of the input where any value less than zero is set to zero. The choice of the ReLU activation facilitates faster training due to the simplicity of its function and derivative evaluations.
3) SoftMax layer is often used in the final layer of a neural network-based classifier. It takes a vector of arbitrary real-valued scores (in z) and reduces it to a vector of values between zero and one that sums to one.

$$z^{[L]} = W^{[L]} a^{[L-1]} + b^{[L]} \tag{60}$$

To compute z, SoftMax activation function is needed.

$$t = expz^{[L]} \tag{61}$$

t and $z^{[L]}$ is (2,1) dimensional vector which is related to the good and bad of output.

$$a^{[L]} = \frac{expz^{[L]}}{\sum_{i=1}^{8} t_i} \tag{62}$$

Output a is going to be the vector t but normalized to sum to 1.

$$\mathrm{ReLU(x)} = x^{+} = \max(0, x) \tag{63}$$

8.6 Training and Testing Using CNNs and Prediction Accuracy

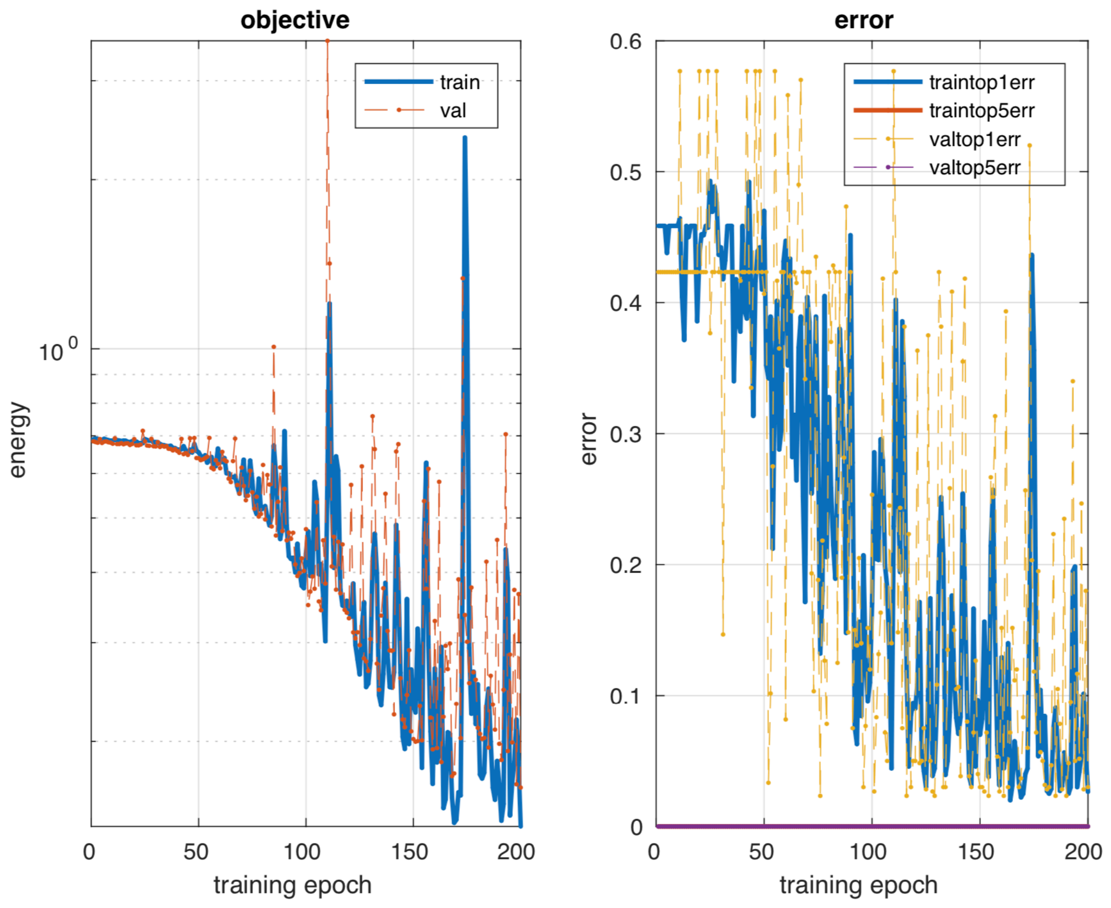


Fig. 8.6. The objective and error plots during the training.

The model for the automatically particle picking has been trained using the MATLAB toolbox Matconvnet. The prediction model has been implemented as a convolution neural network (CNN) with architecture as shown in Fig. 8.5. The proposed convolutional network is composed of two convolutional layers and a fully-connected layer. Each convolutional layer is followed by a max pooling layer and rectified linear unit layer (ReLU). After the two convolution layers, a fully-connected layer with Softmax activation is used in the final layer of convolution neural network-based classifier. In the first convolutional layer, the “weights” are the filters being learned and initialized with random numbers from a Gaussian distribution. The filters are 13 by 13 spatial resolution with 1 filter depth and the number of kernels is 200. The next layer is a max pooling layer which takes 3 by 3 sliding window with a stride of 3.

Table 8.1. CNNs predictions results compared to the test image data and good particles and bad particles represent 1 and 2 respectively.

Test image number : Label	CNNs prediction results (Good and Bad)
Test image 1 : 1 (Good)	1 (Good)
Test image 2 : 1 (Good)	1 (Good)
Test image 3 : 2 (Bad)	2 (Bad)
Test image 4 : 1 (Good)	1 (Good)
Test image 5 : 2 (Bad)	2 (Bad)
Test image 6 : 1 (Good)	1 (Good)
Test image 7 : 2 (Bad)	2 (Bad)
Test image 8 : 2 (Bad)	2 (Bad)
Test image 9 : 1 (Good)	1 (Good)
Test image 10 : 2 (Bad)	2 (Bad)

The spatial resolution will be decreased due to the stride 3 in max pooling layer. The ReLU activation function is then applied to introduce non-linearities into the modeling framework. The last layer is a fully-connected layer which has dense connections with the activation function outputs from the previous convolution layers. The final layer is designed to produce 2 output values corresponding to the model's predicted likelihood for the good and bad particles. Using the CNNs mentioned above with the 2000 training data and the 600 test data, the learning rate is 0.001, the batch size is 50, and the epoch is 200 for the training. In this convolutional neural network, MatConvNet minimizes the objective which represents the loss function and y-axis of energy represents to measure the magnitude of loss. During the training, several statistics are measured after every batch. In the objective and the error plots, the training error is depicted by the blue line and the validation error is represented by an orange dotted line. The error graph should show similarity to the objective graph and our network achieved 0.033 validation error. Lastly, the results are shown in Table. 8.1. The left side shows the number of the 10 test images and the marked label among 600 test images, and the right side shows the prediction based on the training results. As we can see in Table. 8.1, labelled test images and CNNs prediction results are quite match the results each other. To evaluate the accuracy of training data obtained from CNNs, the success rate is defined as

$$\frac{number\ of\ correct\ test\ images}{total\ number\ of\ test\ images} \times 100\% \tag{64}$$

The prediction results of the inverse problem using the training data showed a success rate of 96.7 %.

8.7 Results

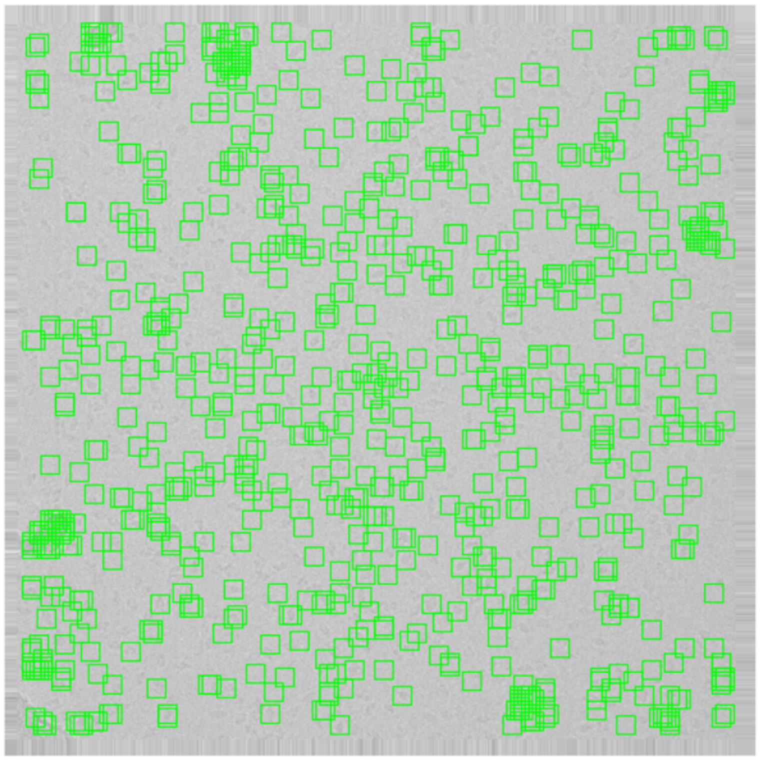


Fig. 8.7. Automatically particle picking in Cryo-EM raw image using trained data.

We have trained using supervised learning to automatically extract proteins in Cryo-EM images. From now on, this training data will be applied to the new Cryo-EM image and the protein particles will be automatically extracted and displayed as a rectangle in the extracted location. To do this, we used MATLAB to create the scan code. The scan code was scanned by moving the raw Cryo-EM image by 10 pixels using training data obtained using supervised learning. During the process, a threshold score is specified. If the score is larger than the threshold, a square is displayed at that position. If the score is less than the threshold score, skip is performed. The result is Fig. 8.7. We have so far obtained training data through supervised learning by designating good and bad particles to pick particles automatically. During the training, CNNs were used and the success rate was 96.7%. Using this training data, protein particles in the Cryo-EM image were extracted through the scan code. In order to increase the accuracy during the process, it is possible to control the threshold score to extract the correct particles and to narrow the range of the scanned pixels when scanning the image.

CHAPTER 9. ANOMALY AND BAD DATA DETECTION FOR POWER SYSTEMS BY LSTMS

9.1 Introduction

Continuous or simultaneous failures / events in power systems are a common problem that can lead to frequent section blackouts. In recent decades, many researches have been reported on perturbation or event detection and perception. Various types of signals were used for analysis purposes, including frequency, power, voltage, and phasor angles. In this paper, we choose to analyze the frequency signal because suddenly the frequency changes in the power system due to events such as generator trip, line trip or load shedding in large power systems. Typical power system events are divided into one of three categories such as generator trip (GT), load shedding (LS), and line trip (LT).

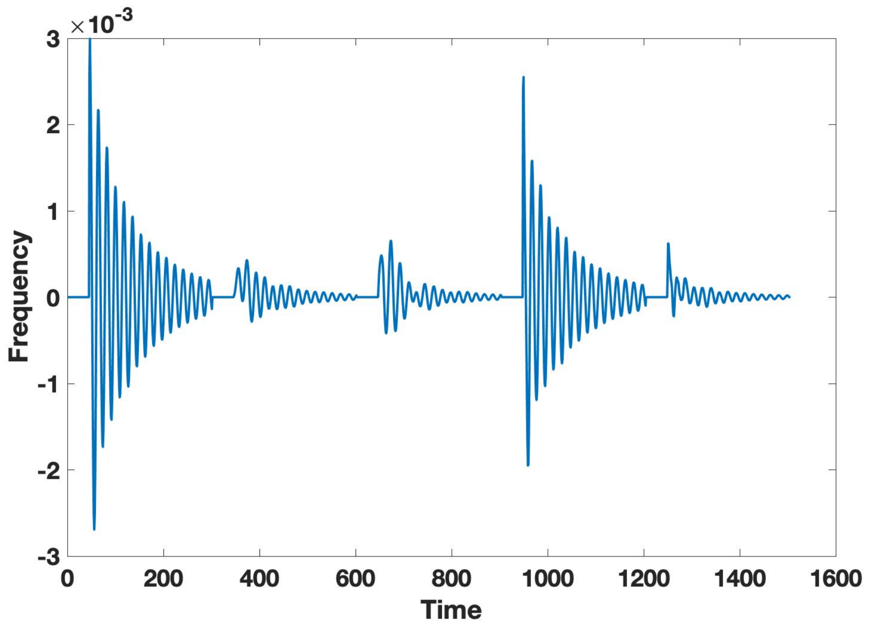


Fig. 9.1. Line trip (LT) data for power system.

For GT and LS, vibration can be treated as noise. In LT, the frequency pattern of vibration is similar to the frequency pattern of LT, and the scale of oscillation is similar to that of LT. Therefore, Detection of LT is a great challenge to the power systems.

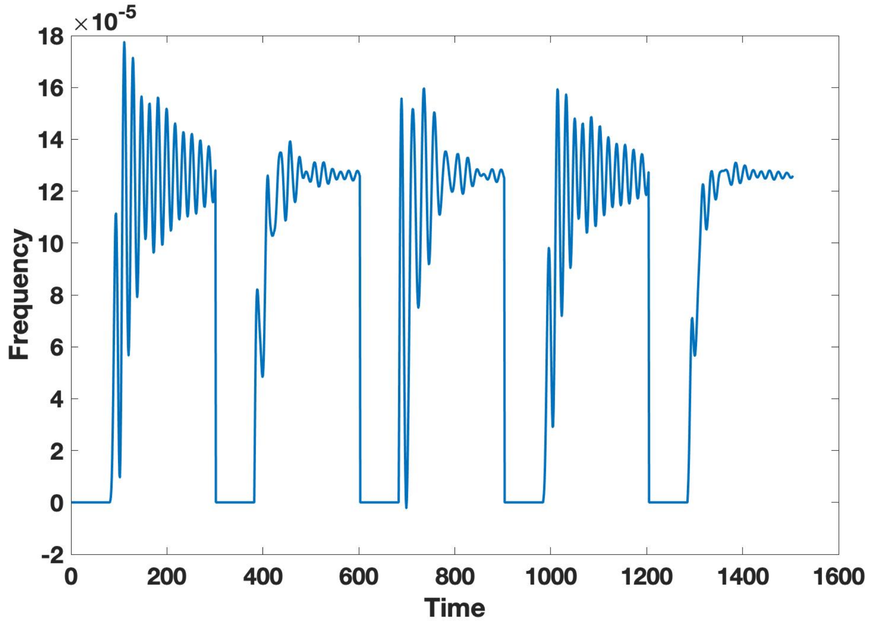


Fig. 9.2. Load shedding (LS) data for power system.

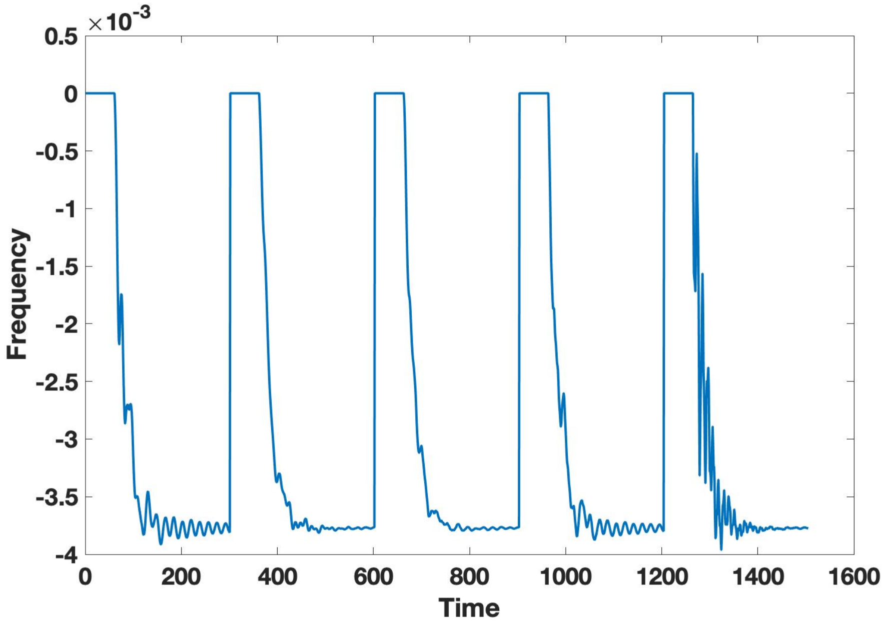


Fig. 9.3. Generator trip (GT) data for power system.

Therefore, in this paper, we used various methods to detect LT and predict the most predictable method. The remainder of this paper is organized as follows: Section 9.3 is introduced the variety of methods to predict line trip data and find the best prediction method. Section 9.4 is shown the results of predicting line trip data using LSTMs and also shows the result of detecting bad data using LSTMs by arbitrarily adding bad data to the real data.

9.2 Dataset Creation for Predicting the Line Trip Data

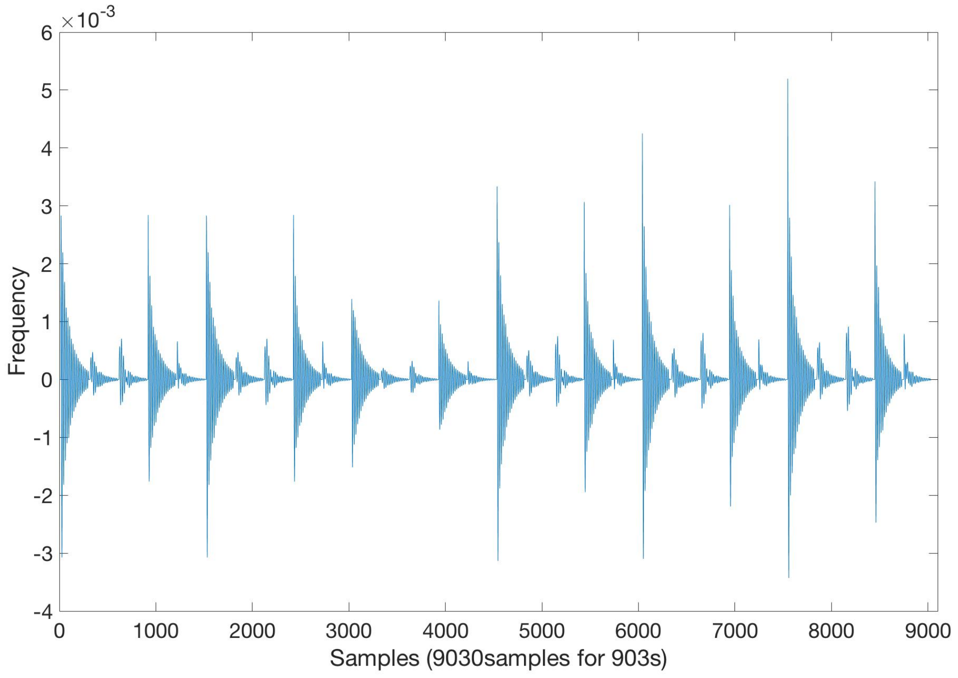


Fig. 9.4. Line trip (LT) data for prediction with LSTMs.

In this section, we introduce data for predicting line trip data using various methods. As shown in Fig. 9.4, the line trip data is frequency data having a certain type of period, and the x-axis represents samples of data. The data obtained 9030 samples in 903 seconds, and the y-axis shows the magnitude of the frequency. Next, we used the measurement data to detect the bad data in the data. As shown in Fig. 9.5, the measurement data is frequency data having a certain periodic form like the line trip data, and the x-axis represents time (s). Measurement data was obtained from one data per second and 902 data were measured for a total of 992 seconds. Also, when compared with line trip data, we used about 10 times less data. The reason is that we use a small number of data to find out if it can be predicted using various methods and use a small number of data to find out the ability to detect bad data.

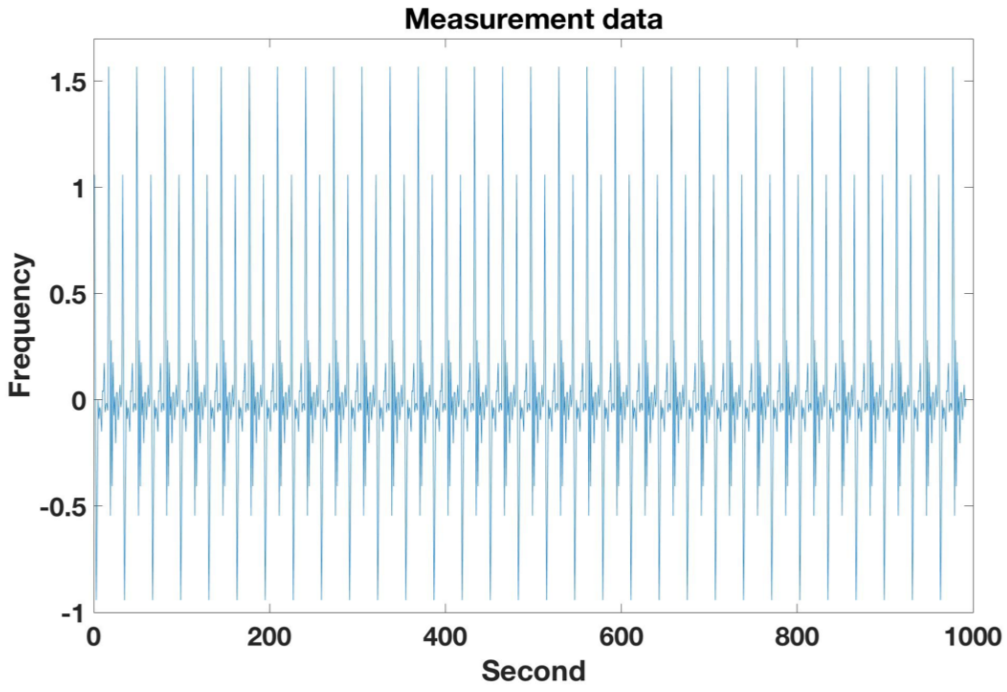


Fig. 9.5. Measurement data for prediction with LSTMs.

9.3 Methods

In this section, we have used four methods to predict the line trip data and the measurement data, and to find the optimal method to detect bad data by arbitrarily adding four outliers to the measurement data. Therefore, we want to find the optimum method by comparing and analysing the Root Mean Square Error (RMSE) using real data and predicted data.

9.3.1 Long Short Term Memory (LSTM)

LSTM [88] is a special kind of RNN as long short term memory networks. The most basic characteristic of LSTM is that it stores information for a long time. Therefore, LSTM can solve the long-term dependency problem of RNN.

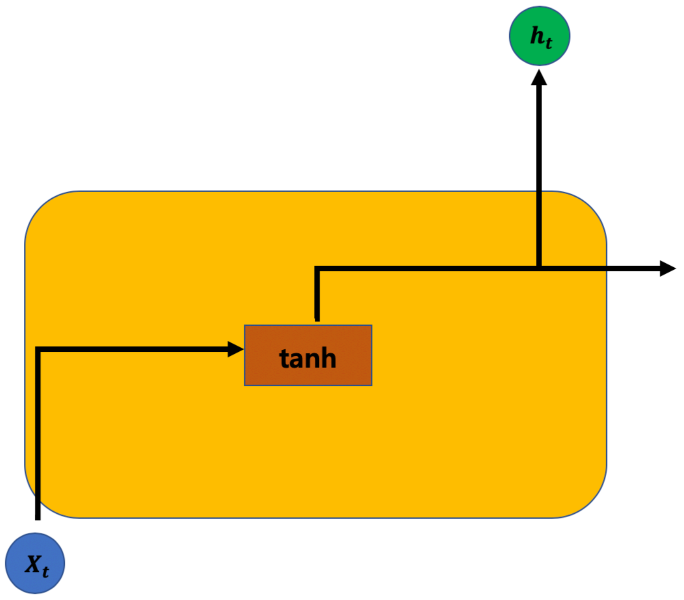


Fig. 9.6. Repeated standard RNN module with single layer.

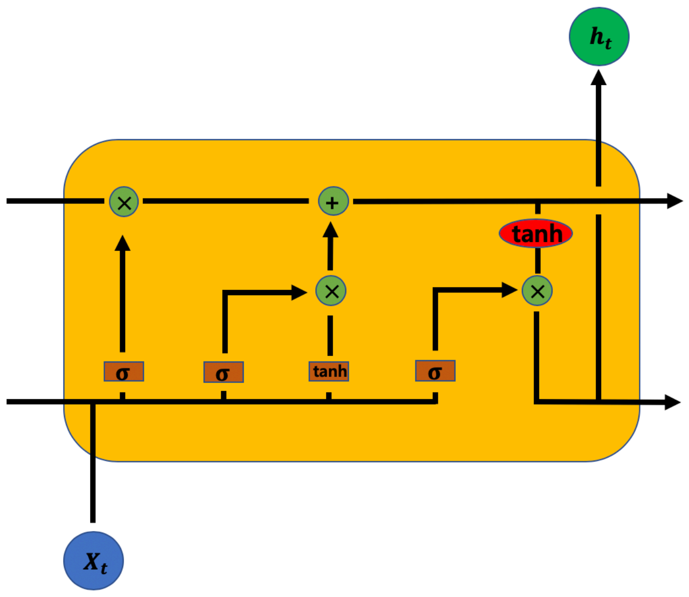


Fig. 9.7. Repeated modules with four interacting layers that the LSTMs contains.

Fig. 9.6 consists of chains that are repeated with the structure of the RNN. X_t and h_t represent the input and output respectively and also contain a single tanh layer as an activation function. Fig. 9.7 represents the repetitive structure of LSTM, and it has a more complicated structure when compared with the structure of RNN. Here, the yellow rectangle represents the neural network layer and the pink circle represents the pointwise operation.

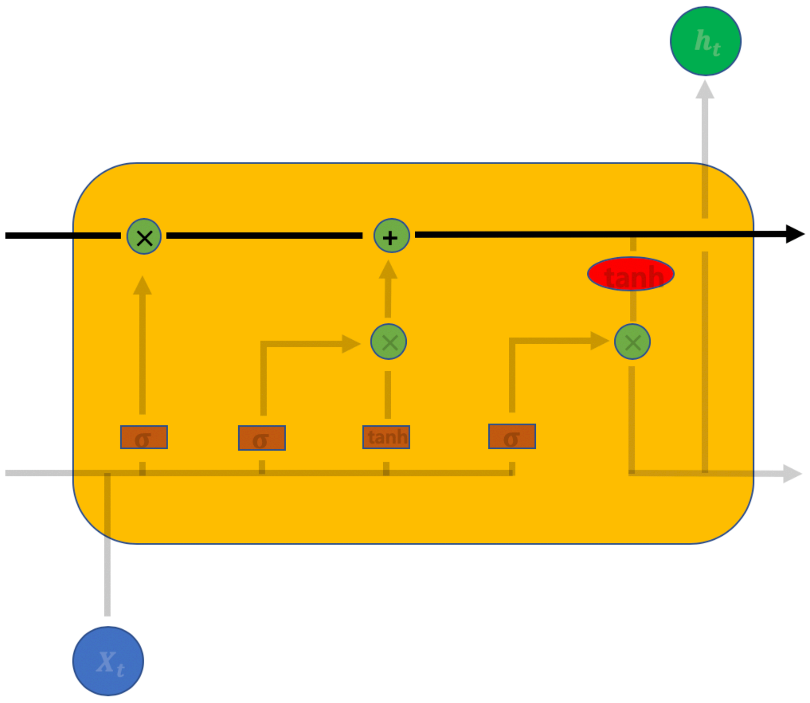


Fig. 9.8. The cell state of the LSTMs.

Fig. 9.8 shows the cell state of LSTM. The cell state is the key point of the LSTM and is the horizontal line at the top of the Fig. 9.8. The cell state is made up of minor linear operations and plays a very important role in transmitting information. It has a function to add or remove information using the gate in the process of transmitting information. So far, we have examined the overall structure of LSTMs and the role of cell states. Next, we will use LSTMs in more detail to see how the information we want flows.

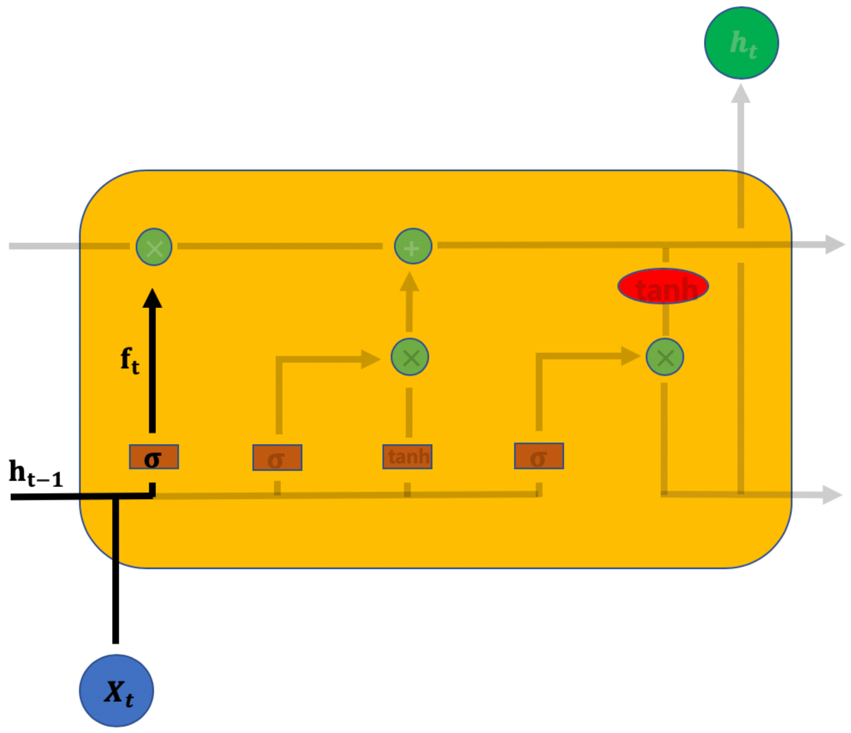


Fig. 9.9. First step of the LSTM with sigmoid layer and inputs.

Fig. 9.9 shows the first step of the LSTM. The first step is to choose what information to discard in the cell state. This process is done by taking input values of $h_{t-1}$and X_t with output values between 0 and 1 in the sigmoid layer which is a forget gate layer. Here, if the output value is 1, the value is completely maintained. If the output value is 0, the value is completely discarded. This process takes the input data through the sigmoid layer and proceeds to the next process by the following equation.

$$f_t = \sigma(W_f \cdot [h_{t-1}, X_t) + b_f) \tag{65}$$

In the equation, h_{t-1} and X_t are taken as inputs in the sigmoid layer, and the product of the weight function and the bias b_f is added to the cell state.

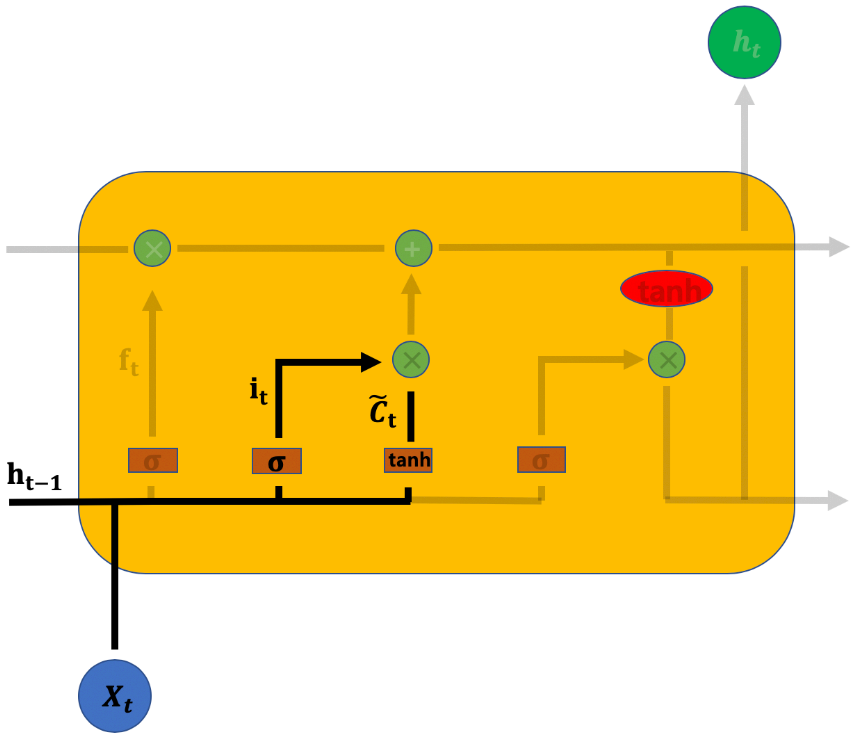


Fig. 9.10. Second step of the LSTM with sigmoid layer and tanh activation function.

The second step determines whether new information is stored in the cell state and is shown in Fig.9.10. First, the sigmoid layer determines which values we update. The following tanh layer produces a new candidate value $\tilde{C}_t$ which can be added to the cell state, and finally the two values to affect the next state. This process is done by the following formula.

$$i_t = \sigma(W_f \cdot [h_{t-1}, X_t) + b_i) \qquad (66)$$

$$\tilde{C}_t = tanh(W_C \cdot [h_{t-1}, X_t) + b_C) \qquad (67)$$

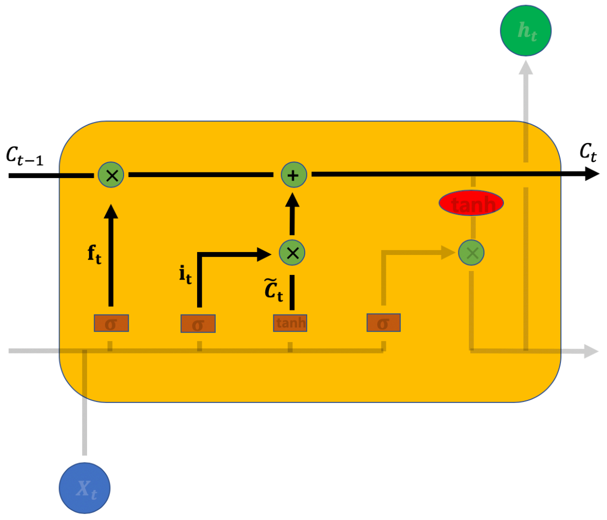


Fig. 9.11. Third step of updating the old cell state to the new cell state.

The third step shown in Fig. 9.11 updates the old cell state C_{t-1} to the new state C_t. This process multiplies the old cell state by f_t obtained in the first step. In addition, the data that is discarded from the forget gate layer is discarded. Then, by adding the product of i_t and $\tilde{C}_t$, the new candidate value affects the existing value.

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{68}$$

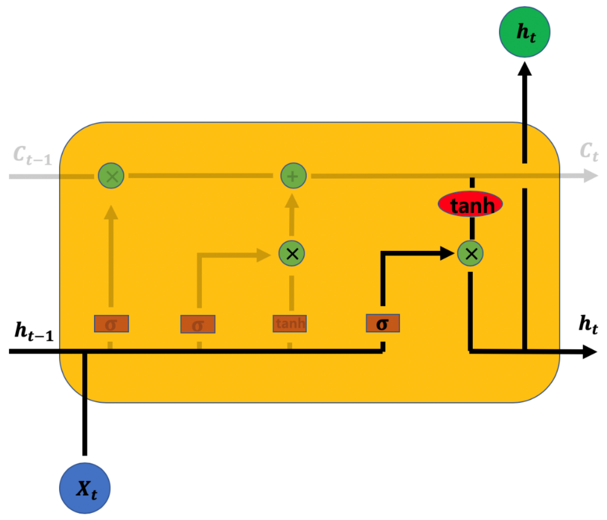


Fig. 9.12. The final step of the LSTMs.

Finally, Fig. 9.12 shows the final step. In this step we have to determine the output value. First, we use a sigmoid layer to determine what value to output. Then, the value between -1 and 1 is obtained by using the tanh activation function. This value can then be multiplied by the output of the sigmoid gate layer to reflect the desired result. The equation is as follows:

$$o_t = \sigma(W_0[h_{t-1}, X_t] + b_0 \tag{69}$$

$$h_t = o_t * \tanh\,(C_t) \tag{70}$$

9.3.2 STLDecompose

STLDecompose [89] is a powerful time series decomposition technique that can be used in a variety of situations. STL stands for "Seasonal and Trend decomposition using Loess". Here Loess is a technique for estimating nonlinear relations. The key idea is that the time series data (D) can be broken down into three components: trending (T), seasonal (S) and residual (R).

$$\mathrm{D} = \mathrm{T} + \mathrm{S} + \mathrm{R} \tag{71}$$

STL has several advantages: STL can handle any kind of seasonality, including monthly or quarterly data. Seasonal ingredients can change over time. User can control the rate of change of seasonal components and adjust the smoothness of the trend - seasonal. Sometimes an ideal value cannot affect the trend and seasonal components, but ideal value will affect the remainder.

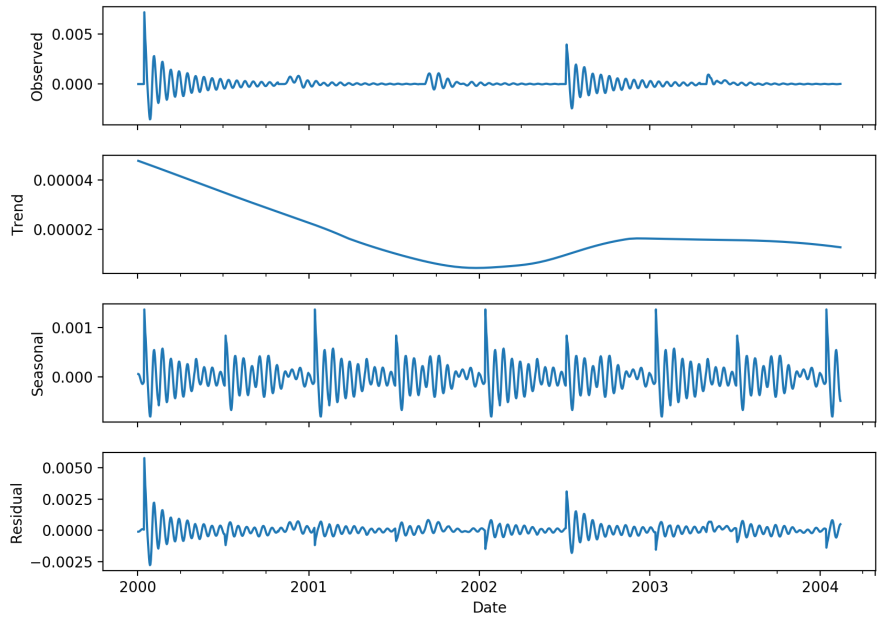


Fig. 9.13. Decomposition for trend, seasonal and residual with line trip data.

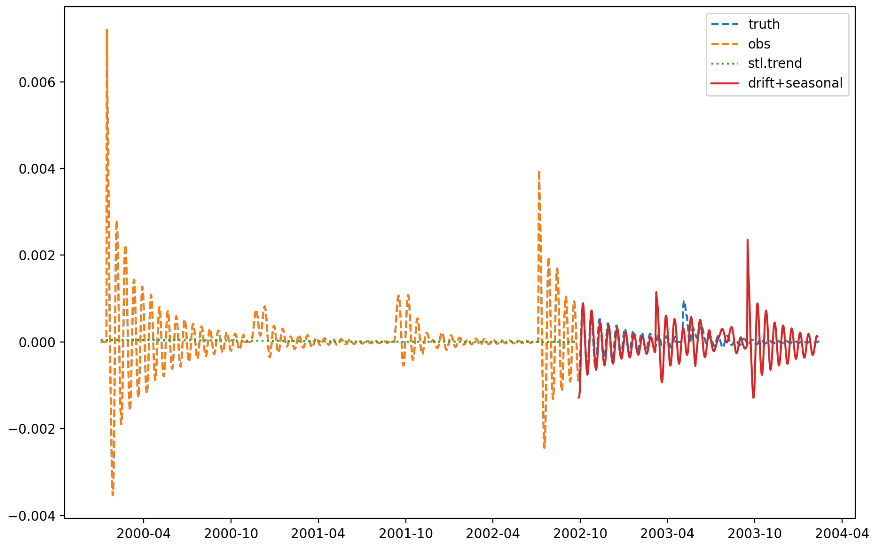


Fig. 9.14. Prediction results for line trip data with STLDecompose.

Fig. 9.13 shows an example of applying STL to the line trip data. As you can see in the figure, using SLTDecompose, we can decompose the data to see the trend and seasonal of the data. Also, as can be seen in Fig. 9.14, we learn the line data from the yellow data such as the observed data to predict future data sufficiently, and the results are quite similar.

9.3.3 ARIMA Model

Three integers (p, d, q) used to parameterize the ARIMA model [90]. These three parameters represent the seasonality, trends, and noise of the data. p is the autoregressive part of the model, which allows us to integrate the influence of past values into our model. d is the integrated part of the model, which includes the term of the model that incorporates the amount of difference to apply to the time series. q is the moving average part of the model that allows us to configure the model's error as the linear combination of error values observed at a previous point in the past. When dealing with seasonal effects, seasonal ARIMA marked as ARIMA (p, d, q) (P, D, Q)s is used. (p, d, q) is the seasonal parameter, (P, D, Q) applies to the seasonal component of the time series and s is the time series period (4 for the quarter and 12 for a year). Three pairs of parameters are used to train with the ARIMA model. When evaluating and comparing statistical models with different parameters, each parameter can be ranked according to how well they fit in the data. We use the Akaike Information Criterion (AIC) value in this process. The AIC statistic model is used to return to the ARIMA model. The AIC values compare how well the model fits the data and take into account the complexity of the model, so a model that fits better while using fewer features will receive a lower AIC score than a similar model that uses more features.

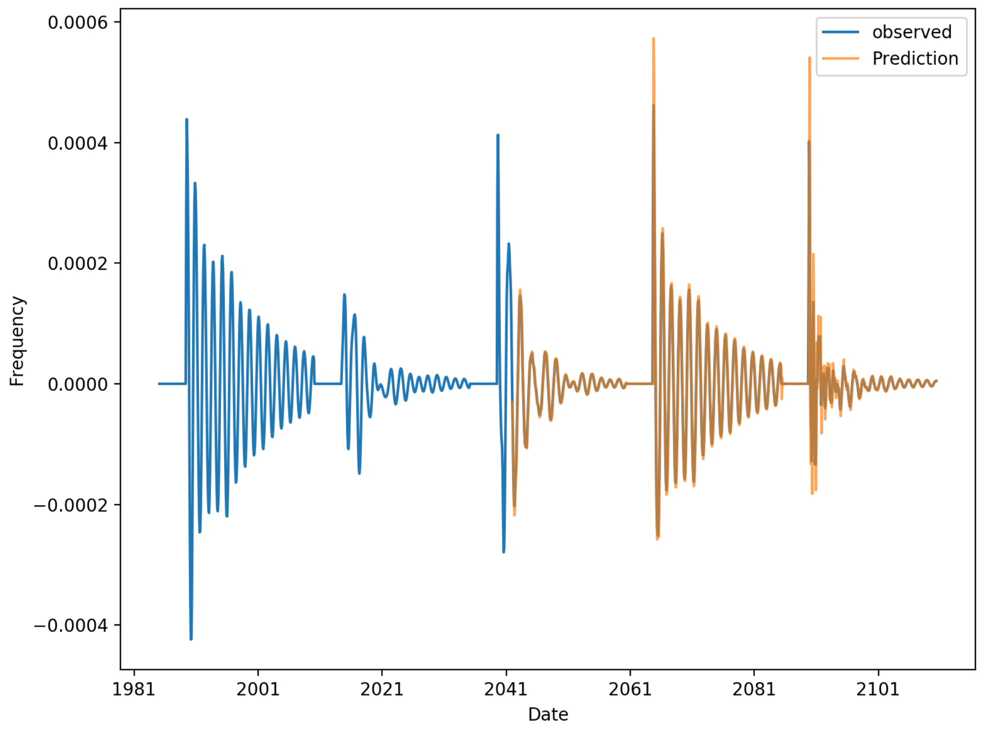


Fig. 9.15. Prediction results for line trip data with ARIMA model.

Table 9.1. Results of AIC (Akaike Information Criterion) score with different parameters (p, d, q) × (P, D, Q, s).

parameters (p, d, q) × (P, D, Q, s)	AIC score
(1, 1, 1) × (0, 0, 0, 12)	-27896
(0, 1, 1) × (0, 0, 0, 12)	-27490
(1, 0, 1) × (0, 0, 0, 12)	-27557
(1, 1, 0) × (0, 0, 0, 12)	-27741
(1, 1, 1) × (1, 0, 0, 12)	-27693
(1, 0, 0) × (1, 0, 0, 12)	-26611
(0, 1, 1) × (1, 0, 1, 12)	-27284
(0, 1, 0) × (1, 1, 0, 12)	-25327
(0, 1, 0) × (1, 0, 0, 12)	-26581
.	.

In this study, the AIC score was obtained by substituting various parameters to find the lowest AIC score mentioned above. The result is shown in Table 9.1. As the table shows, when the parameter is (1, 1, 1) x (0, 0, 0, 12), the AIC score is the lowest value of -27896 and it can be seen in Fig. 9.15. The blue line represents the observed data and the yellow line represents the predicted results using the ARIMA model. As can be seen from the figure, the prediction results are quite matched the line trip data.

9.3.4 Holt-Winters Seasonal Method

Holt-Winters seasonal method [91] is extended the Holt's method to capture seasonality. It consists of the predictive equation and three smooth equations. One for the level l_t, one for the trend b_t and one for the seasonal component s_t, each of them is made up of the corresponding smoothing parameters α, β^*, γ. We use m to denote the period of the seasonality, i.e., the number of seasons in a year. For instance, for quarterly data m=4, and for monthly data m=12. There are two variations on this method, depending on the nature of the seasonal components. Addition method is used when the seasonal variability is almost constant throughout the time series and multiplication method is used when seasonal variation is proportional to the level of the time series. In this research, we used the addition method and the govern equations represents below:

$$\hat{y}_{t+h|t} = l_t + hb_t + s_{t-m+h_m^+} \tag{72}$$

$$l_t = \alpha(y_t - s_{t-m}) + (1-\alpha)(l_{t-1} + b_{t-1}) \tag{73}$$

$$b_t = \beta^*(l_t - l_{t-1}) + (1-\beta^*)b_{t-1} \tag{74}$$

$$s_t = \gamma(y_t - l_{t-1} - b_{t-1}) + (1-\gamma)s_{t-m} \tag{75}$$

The level equation shows the weighted average between the seasonally adjusted observation $(y_t - s_{t-m})$ and the non-seasonal forecast $(l_{t-1} + b_{t-1})$ for time t. The trend equation is identical to Holt’s linear method. The seasonal equation shows the weighted average between the current seasonal index, $(y_t - l_{t-1} - b_{t-1})$ and the seasonal index of the same season last year (i.e., m time periods ago).

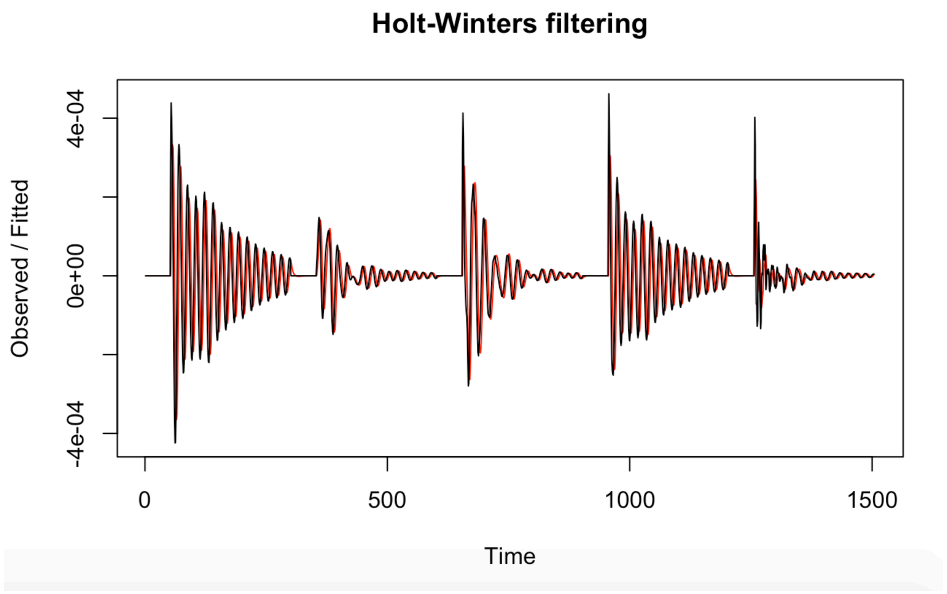


Fig. 9.16. Prediction results for line trip data with Holt-Winters method.

In Fig. 9.16, we learned the line data using Holt-Winters methods to predict data sufficiently, and the results are quite similar compared to line trip data.

9.3.5 The Comparison of the Prediction Data with Methods

Table 9.2. RMSE comparison for 4 methods.

Methods	RMSE(Root Mean Square Error)
Long Short Term Memory (LSTM)	0.002605
Seasonal Trend decomposition using Loess (STLD)	0.0381
AutoregRessive Integrated Moving Average (ARIMA)	0.00311
Holt-winters seasonal method	0.019

We have predicted line trip data using 4 methods that we have briefly discussed above. In order to find the most accurate prediction method, Root Mean Square Error (RMSE) was calculated and compared. Table. 9.1 represents the result of comparing RMSE values using four methodologies. As can be seen from the table, it shows the smallest error value when the line trip data is predicted using LSTM. Therefore, in the next section, we will use LSTM to predict line trip data and add 4 outliers to measurement data to find bad data and to predict measurement data.

9.4 Predicting the Line Trip Data and Bad Data Detection with LSTMs

In this section, the methodology of LSTMs was deemed to be most appropriate when using the line trip data and measurement data and evaluating the methods mentioned above as prediction methods. Therefore, we will go into more detail in this section.

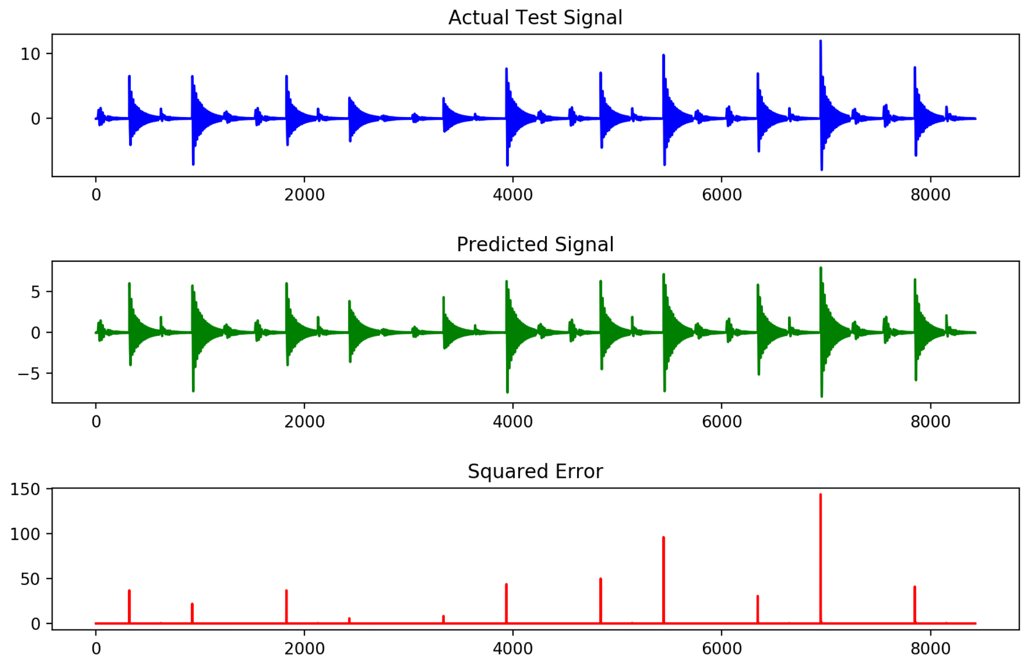


Fig. 9.17. Line trip data for prediction with LSTMs.

We also compared the real data with the prediction data using LSTMs by arbitrarily including outlier data in the measurement data. We want to see the result of detecting bad data through the obtained error. We first obtained the 9030 line trip data [92] for 903 seconds. Using 9030 data, training data and test data are composed of 6030 and 3000 respectively and 900 validation data sets are specified in 6030 training data sets. We trained using the specified data and LSTMs. During the training process, the batch size is set to 50 and the epoch is set to 100. The LSTM layers start from the input layer 1 and through the hidden layers 64, 256, and 100, the final output layer is 1. We predicted the data using the mentioned above data and LSTMs. Fig. 9.17 shows the result of predicting line trip data using LSTMs. The x-axis represents data over time and the y-axis represents the frequency magnitude. The y-axis of the squared error represents the error value obtained by subtracting the predicted signal from the actual test signal. As shown in the figure, the blue data at the top represents the actual test signal data, the data using the mentioned above data and LSTMs. Fig. 9.17 shows the result of predicting line trip data using LSTMs. The x-axis represents data over time and the y-axis represents the frequency magnitude. The y-axis of the squared error represents the error value obtained by subtracting the predicted signal from the actual test signal. As shown in the figure, the blue data at the top represents the actual test signal data, the middle represents the predicted value using LSTMs, and the bottom graph shows the error value obtained by subtracting the predicted signal data value from the actual test signal data. As can be seen from the figure, the predicted results are quite similar when compared to the actual test data.

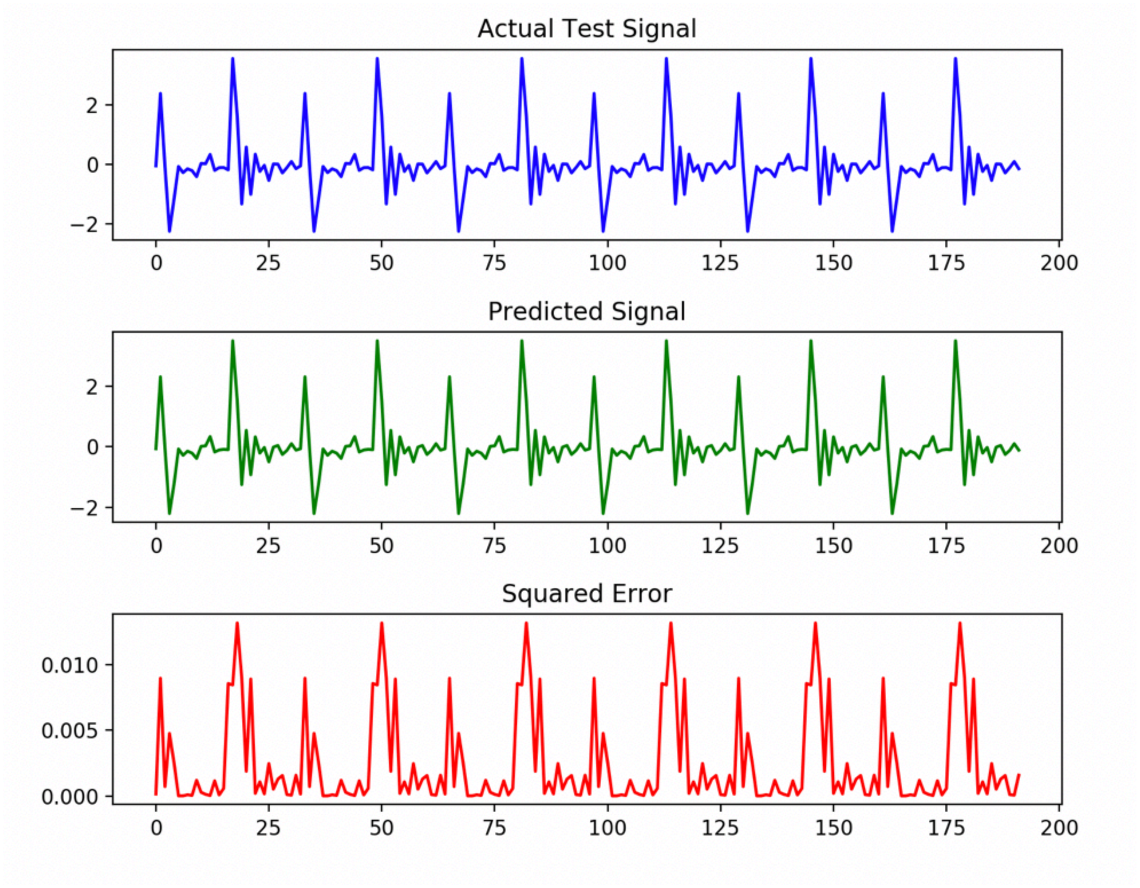


Fig. 9.18. Measurement data for prediction with LSTMs.

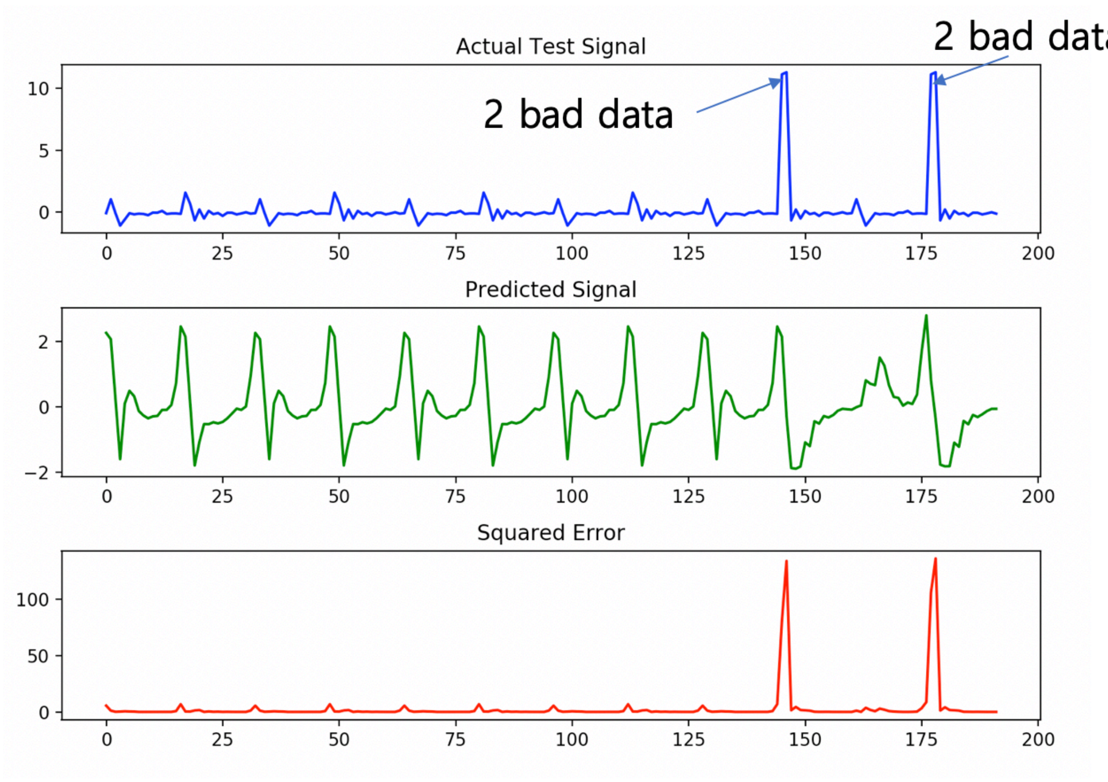


Fig. 9.19. Bad data detection with LSTMs.

Next, we predicted the data using measurement data and LSTMs. Through the total of 992 data, 800 training data and 192 test data were specified. Finally, 60 validation data sets were specified in 800 training data. We trained using the specified data and LSTMs. During the training process, the batch size is set to 50 and the epoch is set to 100. The LSTM layers start from the input layer 1 and through the hidden layers 64, 256, and 100, the output layer is 1. We predicted the measurement data using the mentioned above the data and LSTMs. In addition, as mentioned above, when a small amount of data is used, the amount of data which is 10 times smaller than the number of line trip data is specified in order to judge how high the success rate of data is predicted. Fig. 9.18 shows the result of predicting measurement data using LSTMs. The x-axis represents data over time and the y-axis represents the frequency magnitude. The y-axis of the squared error represents the error value obtained by subtracting the predicted signal from the actual test signal. As shown in the figure, the blue data at the top represents the actual test signal data, the middle represents the result predicted using LSTMs, and finally the bottom graph shows the error value obtained by subtracting the predicted signal data value from the actual test signal data. As can be seen from the graph, the predicted results are quite similar when compared to actual test data.

Finally, bad data was detected using the same measurement data as above. During this process, 4 outliers were arbitrarily added to the data to specify bad data. Fig. 9.19 shows the result of predicting bad data detection of measurement data using LSTMs. As shown in the figure, the blue data at the top represents the actual test signal data, the middle represents the result predicted using LSTMs, and finally the bottom graph shows the error value obtained by subtracting the predicted signal data value from the actual test signal data. As can be seen from the graph, the predicted results are quite similar when compared with the actual test data, and the prediction is different when the outlier part is met. In order to obtain a more accurate result, the errors are calculated and it is confirmed that the error increases rapidly in the four outlier sections. Therefore, we could detect bad data of measurement data through the error.

CHAPTER 10. CONCLUSIONS

In this study, we started from the theoretical part of molecular dynamics and supplemented it by using machine learning and deep learning to compensate for the disadvantages of the high computational cost of molecular dynamics. Also, peridynamics based on molecular dynamics was used as data, and deep learning method was applied to increase the efficiency of computation cost. The deep learning method was used to automatically extract particles from the Cryo-EM image and apply power system frequency data to predict the data and detect bad data.

First, Tersoff empirical potentials, which are the basis of molecular dynamics calculations, are not suitable for calculating the mechanical properties of silicon nanowires. Therefore, we can obtain the mechanical properties suitable for the silicon nanowires by fitting the parameters of the existing experiential potential in our way through the combination of first principle calculation and MD calculation. In the application part study of molecular dynamics, the MEAM potential was applied to the Fe-Cu system to improve the mechanical properties. In the Fe-Cu system, Cu-steel with lamellae Cu precipitates, consisting of 10% Cu in structure, Cu-steel with spherical Cu precipitates, including 5% of randomly mixed Cu, and steel without Cu-containing are compared. The experimental finding is validated by the molecular dynamical simulations, which confirmed that changing the morphologies of Cu precipitates and the ratio of multi-phases in the microstructure can significantly enhance both ultra-high-strength and good ductility.

Secondly, we compensate for the disadvantages of FEM by using peridynamics based on molecular dynamics and also supplemented the disadvantages of calculation of molecular dynamics by applying deep learning technique. In this work, we introduce convolutional neural networks designed to predict and analyze damage patterns on a disk resulting from molecular dynamics (MD) collision simulations. The simulations under consideration are specifically designed to produce cracks on the disk and, accordingly, numerical methods which require partial derivative information, such as finite element analysis, are not applicable. These simulations can, however, be carried out using peridynamics, a nonlocal extension of classical continuum mechanics based on integral equations which overcome the difficulties in modeling deformation discontinuities. Though this nonlocal extension provides a highly accurate model for the MD simulations, the computational complexity and corresponding run times increase significantly as the simulations grow large. We propose the use of neural network approximations to complement peridynamic simulations by providing quick estimates which maintain much of the accuracy of the full simulations while reducing simulation times by a factor of 1,500. We propose two distinct convolutional neural networks: one trained to perform the forward problem of predicting the damage pattern on a disk provided the location of a colliding object's impact, and another trained to solve the inverse problem of identifying the collision location, angle, velocity, and size given the resulting damage pattern. Finally, neural processes were used to solve the 2-D regression problem. These neural processes were optimized to address the regression problem by combining the advantages of neural networks and Gaussian processes. In this process, the data obtained through the peridynamics were made similar to the MNIST data, and were classified into training data set and test data set for training. We also set the context points to 10, 100, 300, and 784 so that we could see the results of the test images. As a result, if the epoch is low, the prediction is not good enough regardless of context points. However, as the number of epochs increases, the result of the speculation is right. In the final epoch 3000, even if the context points are 100, the prediction is sufficiently right as in the case of the context points 784. In other words, using neural processes, the forecast is possible using data with less information.

Finally, deep learning was used to automatically extract the particles in the Cryo-EM image, predict frequency data of the power system, and detect bad data. First, we used CNNs to distinguish between bad data and good data of a particle, and the success rate was 96.7%. Using the trained data, we could create a scanning system and extract the particles by applying a new raw Cryo-EM image. We also used LSTMs, a unique method of RNN, to predict frequency data of the power system. In addition, bad data was predicted by randomly adding outlier to the frequency data. As a result of prediction, if bad data existed, it could be detected by showing that the MSE error value significantly increased, unlike other standard data.

REFERENCES

[1] Sinnott, S. B., & Brenner, D. W. (2012). Three decades of many-body potentials in materials research. *Mrs Bulletin*, *37*(5), 469-473.

[2] Michalak, D. J., Amy, S. R., Aureau, D., Dai, M., Estève, A., & Chabal, Y. J. (2010). Nanopatterning Si (111) surfaces as a selective surface-chemistry route. *Nature materials*, *9*(3), 266.

[3] Que, J. Z., Radny, M. W., Smith, P. V., & Dyson, A. J. (2000). Application of the extended Brenner potential to the Si (111) 7× 7: H system I: cluster calculations. *Surface science*, *444*(1-3), 123-139.

[4] Lee, B., & Rudd, R. E. (2007). First-principles study of the Young's modulus of Si⟨ 001⟩ nanowires. *Physical review B*, *75*(4), 041305.

[5] Eremets, M. I., Trojan, I. A., Medvedev, S. A., Tse, J. S., & Yao, Y. (2008). Superconductivity in hydrogen dominant materials: Silane. *Science*, *319*(5869), 1506-1509.

[6] Reed, G. T., Mashanovich, G., Gardes, F. Y., & Thomson, D. J. (2010). Silicon optical modulators. *Nature photonics*, *4*(8), 518.

[7] Rojas-Lopez, M., Orduña-Diaz, A., Delgado-Macuil, R., Gayou, V. L., Bibbins-Martínez, M., Torres-Jácome, A., & Treviño-Palacios, C. G. (2010). a-Si: H crystallization from isothermal annealing and its dependence on the substrate used. *Materials Science and Engineering: B*, *174*(1-3), 137-140.

[8] Stillinger, F. H., & Weber, T. A. (1986). Erratum: Computer simulation of local order in condensed phases of silicon [Phys. Rev. B 31, 5262 (1985)]. *Physical Review B*, *33*(2), 1451.

[9] Tersoff, J. (1988). Empirical interatomic potential for silicon with improved elastic properties. *Physical Review B*, *38*(14), 9902.

[10] Baskes, M. I. (1992). Modified embedded-atom potentials for cubic materials and impurities. *Physical review B*, *46*(5), 2727.

[11] Lenosky, T. J., Sadigh, B., Alonso, E., Bulatov, V. V., de la Rubia, T. D., Kim, J., ... & Kress, J. D. (2000). Highly optimized empirical potential model of silicon. *Modelling and Simulation in Materials Science and Engineering*, *8*(6), 825.

[12] Murty, M. R., & Atwater, H. A. (1995). Empirical interatomic potential for Si-H interactions. *Physical Review B*, *51*(8), 4889.

[13] U., H., and P., V., 1998. “Hydrogen passivation of silicon surfaces: A classical molecular-dynamics study”. Phys. Rev. B, 57, p. 13285.

[14] S., I., Y., S., S., H., and S., S., 2004. “Development of a molecular dynamics potential for si-h systems and its application to cvd reaction processes”. Surf. Sci., 560, p. 1.

[15] Kresse, G., & Furthmüller, J. (1996). Efficient iterative schemes for ab initio total-energy calculations using a plane-wave basis set. *Physical review B*, *54*(16), 11169.

[16] Yang, M., Pan, Y., Yuan, F., Zhu, Y., & Wu, X. (2016). Back stress strengthening and strain hardening in gradient structure. *Materials Research Letters*, *4*(3), 145-151.

[17] Shen, Y. F., Wang, C. M., & Sun, X. (2011). A micro-alloyed ferritic steel strengthened by nanoscale precipitates. *Materials Science and Engineering: A*, *528*(28), 8150-8156.

[18] Grässel, O., Krüger, L., Frommeyer, G., & Meyer, L. W. (2000). High strength Fe–Mn–(Al, Si) TRIP/TWIP steels development—properties—application. *International Journal of plasticity*, *16*(10-11), 1391-1409.

[19] Jacques, P. J. (2004). Transformation-induced plasticity for high strength formable steels. *Current Opinion in Solid State and Materials Science*, *8*(3-4), 259-265.

[20] Speer, J. G., Edmonds, D. V., Rizzo, F. C., & Matlock, D. K. (2004). Partitioning of carbon from supersaturated plates of ferrite, with application to steel processing and fundamentals of the bainite transformation. *Current Opinion in Solid State and Materials Science*, *8*(3-4), 219-237.

[21] Bouquerel, J., Verbeken, K., & De Cooman, B. C. (2006). Microstructure-based model for the static mechanical behaviour of multiphase steels. *Acta Materialia*, *54*(6), 1443-1456.

[22] Edmonds, D. V., He, K., Rizzo, F. C., De Cooman, B. C., Matlock, D. K., & Speer, J. G. (2006). Quenching and partitioning martensite—A novel steel heat treatment. *Materials Science and Engineering: A*, *438*, 25-34.

[23] Seo, E. J., Cho, L., Estrin, Y., & De Cooman, B. C. (2016). Microstructure-mechanical properties relationships for quenching and partitioning (Q&P) processed steel. *Acta Materialia*, *113*, 124-139.

[24] Seleson, P., Parks, M., Gunzburger, M., & Lehoucq, R. (2009). Peridynamics as an Upscaling of Molecular Dynamics. Multiscale Modeling & Simulation, 8(1), 204-227.

[25] Silling, S., Epton, A., Weckner, M., Xu, O., & Askari, J. (2007). Peridynamic States and Constitutive Modeling. Journal of Elasticity, 88(2), 151-184.

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

[27] Silling, S., & Askari, E. (2005). A meshfree method based on the peridynamic model of solid mechanics. Proposed for Publication in Computers and Structures., 83(17-18), Proposed for publication in Computers and Structures., 2005, Vol.83(17-18).

[28] Bobaru, F., Silling, S. A., & Jiang, H. (2005). Peridynamic fracture and damage modeling of membranes and nanofiber networks. In XI Int. Conf. Fract., Turin, Italy.

[29] Askari, E., Xu, J., & Silling, S. (2006). Peridynamic analysis of damage and failure in composites. In 44th AIAA aerospace sciences meeting and exhibit (p. 88).

[30] Kilic, B., Madenci, E., & Ambur, D. (2006). Analysis of brazed single-lap joints using the peridynamics theory. In 47th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference 14th AIAA/ASME/AHS Adaptive Structures Conference 7th (p. 2267).

[31] Oterkus, S., & Madenci, E. (2017). Peridynamic modeling of fuel pellet cracking. *Engineering Fracture Mechanics*, *176*, 23-37.

[32] Taylor, M., Gözen, I., Patel, S., Jesorka, A., & Bertoldi, K. (2016). Peridynamic modeling of ruptures in biomembranes. *PloS one*, *11*(11), e0165947.

[33] Nikabdullah, N., Azizi, M. A., Alebrahim, R., & Singh, S. S. K. (2014, June). The application of peridynamic method on prediction of viscoelastic materials behaviour. In *AIP Conference Proceedings* (Vol. 1602, No. 1, pp. 357-363). AIP.

[34] Prakash, N., & Seidel, G. D. (2016). Electromechanical peridynamics modeling of piezoresistive response of carbon nanotube nanocomposites. *Computational Materials Science*, *113*, 154-170.

[35] Platt, P., Mella, R., DeMaio, W., Preuss, M., & Wenman, M. R. (2017). Peridynamic simulations of the tetragonal to monoclinic phase transformation in zirconium dioxide. *Computational Materials Science*, *140*, 322-333.

[36] Lall, P., Shantaram, S., & Panchagade, D. (2010). Peridynamic-models using finite elements for shock and vibration reliability of leadfree electronics. Thermal and Thermomechanical Phenomena in Electronic Systems (ITherm), 2010 12th IEEE Intersociety Conference on, 1-12.

[37] Ahadi, A., Hansson, P., & Melin, S. (2016). Indentation of thin copper film using molecular dynamics and peridynamics. *Procedia Structural Integrity*, *2*, 1343-1350.

[38] Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

[39] Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167.

[40] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A. (2015, June). Going deeper with convolutions. Cvpr.

[41] Zeiler, M. D., & Fergus, R. (2014, September). Visualizing and understanding convolutional networks. In European conference on computer vision (pp. 818-833). Springer, Cham.

[42] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. nature, 521(7553), 436.

[43] Lee, B., & Rudd, R. E., (2007). First-principles calculation of mechanical properties of si<001> nanowires and comparison to nanomechanical theory. Phys. Rev. B, 75, p. 194328.

[44] Rasmussen, C. E., & Williams, C. K. (2006). Gaussian processes for machine learning. Number ISBN 0-262-18253-X.

[45] Parussini, L., Venturi, D., Perdikaris, P., & Karniadakis, G. (2017). Multi-fidelity Gaussian process regression for prediction of random fields. Journal Of Computational Physics, 33636-50. doi:10.1016/j.jcp.2017.01.047

[46] Perdikaris, P., & Karniadakis, G. E. (2016). Model inversion via multi-fidelity Bayesian optimization: a new paradigm for parameter estimation in haemodynamics, and beyond. Journal of The Royal Society Interface, 13(118), 20151107.

[47] Perdikaris, P., Venturi, D., Royset, J. O., & Karniadakis, G. E. (2015, July). Multi-fidelity modelling via recursive co-kriging and Gaussian–Markov random fields. In Proc. R. Soc. A (Vol. 471, No. 2179, p. 20150018). The Royal Society.

[48] Raissi, M., & Karniadakis, G. (2016). Deep multi-fidelity Gaussian processes. arXiv preprint arXiv:1604.07484.

[49] Pang, G., Perdikaris, P., Cai, W., & Karniadakis, G. E. (2017). Discovering variable fractional orders of advection–dispersion equations from field data using multi-fidelity Bayesian optimization. Journal of Computational Physics, 348, 694-714.

[50] Perdikaris, P., Venturi, D., & Karniadakis, G. E. (2016). Multifidelity information fusion algorithms for high-dimensional systems and massive data sets. SIAM Journal on Scientific Computing, 38(4), B521-B538.

[51] McKinnon, K. I. (1998). Convergence of the Nelder--Mead Simplex Method to a Nonstationary Point. *SIAM Journal on Optimization*, *9*(1), 148-158.

[52] Lagarias, J. C., Reeds, J. A., Wright, M. H., and Wright, P. E., (1998). Convergence properties of the nelder-mead simplex method in low dimensions. SIAM Journal of Optimization, 9, pp. 112–147.

[53] Broyden, C. G., (1970). The convergence of a class of double-rank minimization algorithms. Journal Inst. Math. Applic., 6.

[54] Fletcher, R., (1970). A new approach to variable metric algorithms. Computer Journal, 13, pp. 317–322.

[55] Goldfarb, D., (1970). A family of variable metric up- dates derived by variational means. Mathematics of Computing, 24, pp. 23–26.

[56] Shanno, D. F., (1970). Conditioning of quasi-newton methods for function minimization. Mathematics of Computing, 24, pp. 647–656.

[57] Coleman, T. F., and Li., Y., (1996). An interior, trust region approach for nonlinear minimization subject to bounds. SIAM Journal on Optimization, 6, pp. 418– 445.

[58] Coleman, T. F., and Li., Y., (1994). On the convergence of reflective newton methods for large-scale nonlinear minimization subject to bounds. Mathematical Programming, 67, pp. 189–224.

[59] Lee, B. J., Wirth, B. D., Shim, J. H., Kwon, J., Kwon, S. C., & Hong, J. H. (2005). Modified embedded-atom method interatomic potential for the Fe− Cu alloy system and cascade simulations on pure Fe and Fe− Cu alloys. *Physical Review B*, *71*(18), 184205.

[60] Daw, M. S., & Baskes, M. I. (1984). Embedded-atom method: Derivation and application to impurities, surfaces, and other defects in metals. *Physical Review B*, *29*(12), 6443.

[61] Baskes, M. I. (1992). Modified embedded-atom potentials for cubic materials and impurities. *Physical review B*, *46*(5), 2727.

[62] Baskes, M. I. (1997). Determination of modified embedded atom method parameters for nickel. *Materials Chemistry and Physics*, *50*(2), 152-158.

[63] Momma, K., & Izumi, F. (2011). VESTA 3 for three-dimensional visualization of crystal, volumetric and morphology data. *Journal of applied crystallography*, *44*(6), 1272-1276.

[64] Plimpton, S. (1995). Fast parallel algorithms for short-range molecular dynamics. *Journal of computational physics*, *117*(1), 1-19.

[65] Lee, B. J. (2006). A modified embedded-atom method interatomic potential for the Fe–C system. *Acta materialia*, *54*(3), 701-711.

[66] Bagaryatsky, Y. A. (1950). Likely mechanism for the tempering of martensite. *Dokl Aksd Nauk SSSR*, *73*, 1161-1164.

[67] Kitware Inc., ParaView web page. http://www.paraview.org/.

[68] Plimpton, S. Pizza.py http://www.cs.sandia.gov/~sjplimp/pizza.html.

[69] Kotsiantis, S. B., Zaharakis, I., & Pintelas, P. (2007). Supervised machine learning: A review of classification techniques. *Emerging artificial intelligence applications in computer engineering*, *160*, 3-24.

[70] Vedaldi, A., & Lenc, K. (2015, October). Matconvnet: Convolutional neural networks for matlab. In *Proceedings of the 23rd ACM international conference on Multimedia* (pp. 689-692). ACM.

[71] Ercolessi, F., & Adams, J. B. (1994). Interatomic potentials from first-principles calculations: the force-matching method. EPL (Europhysics Letters), 26(8), 583.

[72] Balamane, H., Halicioglu, T., & Tiller, W. A. (1992). Comparative study of silicon empirical interatomic potentials. Physical Review B, 46(4), 2250.

[73] Heaton, J. (2017). Ian Goodfellow, Yoshua Bengio, and Aaron Courville: Deep learning.

[74] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems (pp. 1097-1105).

[75] Deng, J., Dong, W., Socher, R., Li, L. J., Li, K., & Fei-Fei, L. (2009, June). Imagenet: A large-scale hierarchical image database. In Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on (pp. 248-255). IEEE.

[76] Yang, X., De Carlo, F., Phatak, C., & Gürsoy, D. (2017). A convolutional neural network approach to calibrating the rotation axis for X-ray computed tomography. Journal of synchrotron radiation, 24(2), 469-475.

[77] Parks, M. L., Littlewood, D. J., Mitchell, J. A., & Silling, S. A. (2012). Peridigm Users' Guide v1. 0.0. *SAND Report*, *7800*.

[78] Parks, M. L., Seleson, P., Plimpton, S. J., Silling, S. A., & Lehoucq, R. B. (2011). Peridynamics with lammps: A user guide, v0. 3 beta. Sandia Report (2011–8253), 3532.

[79] Mitchell, J. A. (2011). A non-local, ordinary-state-based viscoelasticity model for peridynamics. *Sandia National Lab Report*, *8064*, 1-28.

[80] Garnelo, M., Schwarz, J., Rosenbaum, D., Viola, F., Rezende, D. J., Eslami, S. M., & Teh, Y. W. (2018). Neural processes. arXiv preprint arXiv:1807.01622.

[81] Rasmussen, C. E. (2004). Gaussian processes in machine learning. In Advanced lectures on machine learning (pp. 63-71). Springer, Berlin, Heidelberg.

[82] Ahrens, J., Geveci, B., & Law, C. (2005). Paraview: An end-user tool for large data visualization. *The visualization handbook*, *717*.

[83] Ayachit, U. (2015). *The paraview guide: a parallel visualization application*. Kitware, Inc..

[84] Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., ... & Lerer, A. (2017). Automatic differentiation in pytorch.

[85] LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278-2324.

[86] https://www.ebi.ac.uk/pdbe/emdb/empiar/

[87] G. Tang, L. Peng, P.R. Baldwin, D.S. Mann, W. Jiang, I. Rees & S.J. Ludtke. (2007) EMAN2: an extensible image processing suite for electron microscopy. J Struct Biol. 157, 38-46. PMID: 16859925

[88] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735-1780.

[89] Cleveland, R. B., Cleveland, W. S., McRae, J. E., & Terpenning, I. (1990). STL: a seasonal-trend decomposition. *Journal of official statistics*, *6*(1), 3-73.

[90] Zhang, G. P. (2003). Time series forecasting using a hybrid ARIMA and neural network model. *Neurocomputing*, *50*, 159-175.

[91] Chatfield, C., & Yar, M. (1988). Holt-Winters forecasting: some practical issues. *Journal of the Royal Statistical Society: Series D (The Statistician)*, *37*(2), 129-140.

[92] Song, Y., Wang, W., Zhang, Z., Qi, H., & Liu, Y. (2017). Multiple event detection and recognition for large-scale power systems through cluster-based sparse coding. *IEEE Transactions on Power Systems*, *32*(6), 4199-4210.

VITA

Moonseop Kim

School of Mechanical Engineering

Purdue University

EDUCATION

- Ph.D., 2019, School of Mechanical Engineering, Purdue University
- M.S., 2013, Department of Mechanical Engineering, Kyung Hee University
- B.S., 2011, Department of Mechanical Engineering, Kyung Hee University

Research Interests

- Molecular dynamics simulation for materials
- Multi-fidelity Gaussian process regression for multiscale data integration
- Analysis of damage/cracks using peridynamics with deep learning