IDENTIFICATION OF CRACKS IN PIPELINES BASED ON MACHINE LEARNING AND DEEP LEARNING

BY

JINCHEN HE

A THESIS

PRESENTED TO LAKEHEAD UNIVERSITY

IN FULFILLMENT OF THE

THEIS REQUIREMENT FOR THE DEGREE OF

MASTER OF MECHANICAL ENGINEERING

Lakehead University, Faculty of Mechanical Engineering

Thunder Bay, Ontario, CANADA, January 2022

This thesis by Jinchen He is accepted in its present form by the Mechanical Engineering Department of Lakehead University as satisfying the thesis requirements for the degree of Bachelor of Engineering

APPROVED BY

SUPERVISOR

Dr. Hao Bai

Name	**Signature**	**Date**

EXAMINER

Dr. Wilson Wang

Name	**Signature**	**Date**

EXAMINER

Dr. Jian Deng

Name	**Signature**	**Date**

Declaration

I certify that I am the author of this project and that any assistance I received in its preparation is fully acknowledged and disclosed in the project. I have also cited any source from which I used data, ideas, or words, either quoted or directly paraphrased. I also certify that this current study was prepared by me specifically for this course.

No portion of the work referred to in this study has been submitted in support of an application for another degree or qualification to this or any other university or institution of learning

Jinchen He January, 2022

Student Name Signature Date

Abstract

Pipelines are important long-distance transportation structures in modern industry, and because many are buried deep underground, pipeline health monitoring is critical to industry; however, inspecting underground pipelines can be quite challenging due to the large financial and human resources required. For decades, different methods have been used to assess pipeline cracks. Ultrasonic quantitative nondestructive testing (QNDT) is one of the frequently used methods in pipeline health monitoring. In the current study, the coefficients of the reflected and transmitted waves due to different incident waves were first generated by using a semi-analytical finite element method based on classical elasticity theory. In that study, different types of pipes, including different geometries and materials, were considered. Then four different regression machine learning algorithms and three deep learning algorithms were used to identify crack features. In this study, the prediction accuracy was compared between the different algorithms and different datasets. The objective was to find the algorithm with the highest prediction rate and to select a suitable dataset for prediction. It was found that the extremely randomized tree (ERT) algorithm was the best in identifying cracks in the pipeline. The prediction accuracy will be improved by selecting different data sets. In addition, all algorithms performed better in predicting the radial crack depth (CDRD) than predicting the circumferential crack width (CWCD).

Keywords: Non-Destructive Testing, Ultrasonic, Wave Response Coefficient, Machine Learning, Deep Learning

Acknowledgments

My utmost gratitude goes to my thesis supervisor Dr. H. Bai, whose unlimited guidance and dedication have provided a lot of support. It is his continuous encouragement enabled me to complete my thesis research during the epidemic period. He taught me how to carry on the research as clearly as possible, and it was a great privilege to work and learn with him. Meanwhile, I would like to express my deep gratitude to Dr. Wilson Wang and Dr.Jian Deng for their valuable suggestions on my research topic.

Finally, I would like to thank all the people who gave me support during the epidemic, especially my parents, whose unconditional love and care become my strength to facilitate my thesis completion.

Jinchen He

Lakehead University

January 2022

TABLE OF CONTENTS

List of Table 9
List of Figure 10
List of Abbreviations 11
Chapter I Introduction 12
1.1 Literature Review 13
1.1.1 Applications in Mechanical Engineering 13
1.1.2 Applications in Bioengineering 15
1.1.3 Applications in Medical Science 15
1.1.4 Applications in Physics 16
1.1.5 Applications in Other Domain 16
1.2 Outline of the Thesis 17
1.3 Contributions 18
Chapter II Methodology 19
2.1 The Fundamentals of Waves in a Cylinder 19
2.1.1 Equations of Motion 19
2.1.2 Semi Analytical Finite Element 21
2.2 Selection of Data Characteristics 24
2.3 Combination of Machine Learning 25
2.3.1 Determination of Machine Learning Algorithm 25
2.3.2 Principles of Machine Learning Algorithms 25
2.3.2.1 Support Vector Machine (SVM) Algorithm 25
2.3.2.2 Random Forest Algorithm 26
2.3.2.3 Extremely Randomized Tree Algorithm 27
2.3.2.4 K-Nearest Neighbors Algorithm 27
2.3.3 Selection of Hyperparameters in Machine Learning Code 28
2.3.3.1 Support Vector Machines (SVM) 28
2.3.3.2 K-Nearest Neighbors (KNN) 29
2.3.3.3 Random Forest & Extremely Randomized Tree Algorithm 29
2.4 Combination of Deep Learning 30
2.4.1 Determination of Deep Learning Algorithm 30

2.4.2 Principles of Deep Learning Algorithms 31
2.4.2.1 Recurrent Neural Network (RNN) Algorithm Principles 31
2.4.2.2 Principle of Long Short-Term Memory (LSTM) 32
2.4.2.3 Principle of Gated Recurrent Unit (GRU) 35
2.5 Selection of Hyperparameters for Deep Learning Code 36
Chapter III Process and Results 38
3.1 Pre-Setting of Hyperparameters and Pipeline Parameters 38
3.1.1 Settings of Pipeline Parameters 38
3.1.2 Settings of the Machine Learning Hyperparameters 39
3.1.3 Settings of Deep Learning Hyperparameters 41
3.2 Assessment Criteria 42
3.2.1 Mean Absolute Error 42
3.2.2 Coefficient of Determination 42
3.2.3 Root Mean Square Error 43
3.3 Prediction of Pipeline Crack Information 43
3.3.1 Prediction Results of Crack Depth in Radial Direction (CDRD) 45
3.3.2 Prediction Results of Crack Width in Circumferential Direction (CWCD) 47
3.3.3 Results Comparison of CDRD and CWCD 49
3.3.4 Comparison of Machine Learning and Deep Learning 52
3.4 Results & Discussion 55
Chapter IV Conclusion 56
Chapter V Future Works 58
Reference 59
Appendices 62
Appendix A: Control group file about CDRD 62
Appendix B: Control group file about CWCD 64
Appendix C: CDRD prediction codes in Extremely Randomized Tree 66
Appendix D: Part of the CDRD prediction codes in Gated Recurrent Unit 67

List of Table

Fig. 1 Cylindrical Coordinates 19
Fig. 2 Semi Analytical Finite Element in Cylinder..... 21
Fig. 3 Fortran Process 24
Fig. 4 Support Vector Classifier (left) and Support Vector Regression (right)..... 26
Fig. 5 Random Forest Structure Diagram 27
Fig. 6 K-Nearest Neighbors 28
Fig. 7 Example of Recurrent Neural Network (RNN) 31
Fig. 8 Basic structure of a standard recurrent neural network (RNN) 32
Fig. 9 Principle of Long-Short Term Memory (LSTM)..... 33
Fig. 10 Inner structure of a Long Short-Term Memory (LSTM) 34
Fig. 11 Neural Network Figure of Long-Short Term Memory (LSTM)..... 35
Fig. 12 GRU cell..... 36
Fig. 13 CDRD Prediction Histogram..... 47
Fig. 14 CWCD Prediction Histogram..... 48
Fig. 15 Performance comparison histogram of CDRD and CWCD in Control Group..... 49
Fig. 16 Performance comparison histogram of CDRD and CWCD in CFP Group..... 50
Fig. 17 Performance comparison histogram of CDRD and CWCD in TOMR Group 50
Fig. 18 Performance comparison histogram of CDRD and CWCD in ICWN Group 51
Fig. 19 Performance comparison histogram of CDRD and CWCD in PM Group..... 51
Fig. 20 Performance comparison histogram of CDRD and CWCD in Total Group 52
Fig. 21 R^2 comparison histogram 53
Fig. 22 R^2 comparison histogram 54
Fig. 23 GRU loss curve in CDRD prediction..... 54
Fig. 24 GRU loss curve in CWCD prediction..... 55

List of Figure

Table 1 Setting Information about Fortran Code 39
Table 2 Hyperparameters of Machine Learning Algorithms 40
Table 3 Hyperparameters of Deep Learning Algorithms 41
Table 4 Training Data Case 44
Table 5 Using different data types and different algorithms to predict CDRD 45
Table 6 Standard Deviation of Different CDRD Data Types 46
Table 7 Using different data types and different algorithms to predict CWCD 47
Table 8 Standard Deviation of different CWCD data types 48
Table 9 Control group file about CDRD 62
Table 10 Control group file about CWCD 64

List of Abbreviations

Abbreviations	Meaning
CDRD	Crack Depth in the Radial Direction
CWCD	Crack Width in the Circumferential Direction
TOMR	Thickness Over Mean Radius
ICWN	Input Circumferential Wave Number
CFP	Circular Frequency in Pipeline
PM	Pipeline Materials
SVM	Support Vector Machine
KNN	K-Nearest Neighbors
RF	Random Forest
ERT	Extremely Randomized Tree
SRNN	Simple Recurrent Neural Network
LSTM	Long Short-Term Memory
GRU	Gated Recurrent Unit
MAE	Mean Absolute Error
R^2	Coefficient of Determination
CT	Computed Tomography
UT	Ultrasonic Testing
CCTV	Closed-Circuit Television Testing
HMM	Hidden Markov Model
NDT	Non-Destructive Testing
NN	Neural Network
NIR	Near-Infrared Spectroscopy
PCA	Principal Component Analysis
DCNN	Deep Convolution Neural Network
ANN	Artificial Neural Network
CNN	Convolutional Neural Networks

Chapter I Introduction

The rapid development of cities and industries relies heavily on the network of pipelines including multiple applications such as oil pipelines, natural gas transportation, and urban sewage pipeline system. A long-distance natural gas or crude oil pipeline can spread out typically more than 2,000 kilometers, causing a high probability of deterioration/damage occurring to the structure due to either natural disasters or human intervention. In particular, the lifetime of oil pipeline is a crucial part of a few countries' economies, and a ruptured pipeline not only causes significant damage to the economy but also can destroy the ecosystem and environment. For instance, the largest marine oil spill in human history occurred in the Gulf of Mexico on April 20, 2010, resulted in the loss of 11 human lives and $1 billion to the British Petroleum Company. It negatively impacted the fishing economy of Louisiana in the United States. In addition, crude oil pollution caused serious damage to the ecological systems, led by some vulnerable species in extinction.

Therefore, the operation process of such long-distance and large-diameter-based pipelines requires to conduct regular health monitoring in order to safeguard pipelines from incurred damages, namely corrosion and rust, stress deformation, and welding defects. This article primarily outlines the optimization of crack/defect detection methods for pipelines. Among them, methods such as radio-graphic flaw detection, Computed Tomography (CT), Ultrasonic Testing (UT) technology, and Closed-Circuit Television Testing (CCTV) methods are commonly used. However, deep buried pipelines consisting of oil or liquid water are hard to be inspected, for instance, Closed-Circuit Television Testing (CCTV) is particularly inconvenient in such cases due to the limitation of a robotic car not to enter inside the pipeline and take pictures. Conversely, the use of ultrasonic testing technology can meet most testing needs in practice. The principle of ultrasonic detection technology mainly uses the characteristics of ultrasonic waves propagating along the pipeline, which are reflected back from the edge of the crack interface. This information is used to detect and inspect pipeline defects, but this technology relies on specialized equipment such as transducers and proficient experts to classify and judge the images generated by the instrument. Relying on humans to qualitatively evaluate images is undoubtedly very inefficient compared to using neural networks. Many neural networks primarily utilize the whole image or values generated by the ultrasound machine as the basis for neural network

training. However, there is not sufficient data for training. The error in the classification of defects or prediction results is relatively large in this instance. Therefore, in this work, the wave response coefficients generated by Fortran software as a dataset are used and combined with a variety of machine learning and deep learning algorithms for comparison and discussion. This strategy can generate a large amount of data while reducing labor costs.

1.1 Literature Review

1.1.1 Applications in Mechanical Engineering

In 2008, M. Wolff [1] conducted a health examination using the acoustic structure of the components of an aircraft. The aluminum plates and B-CFRP plates with cracks were arranged into two groups i.e., A and B for experimental comparison. The transducer on the aluminum plate and B-CFRP plate were arranged in a circular form and matrix form, respectively. A Hidden Markov Model (HMM) and Support Vector Machine (SVM) model were used as statistical classifiers to classify the plates with or without cracks. They observed that a statistical classifier cannot accurately locate the crack position, while a high degree of accuracy in the classification of crack is achieved. In terms of classification accuracy, the support vector machine (SVM) model was higher than the Hidden Markov Model (HMM), and its classification accuracy of isotropic materials was higher than that of composites.

In 2008, Chengjun Jiang [2] reported detection technologies of pipelines including radiographic inspection, Ultrasonic Testing (UT), metal magnetic test, etc., and explained that Ultrasonic Testing (UT) technology was better than other technologies in detecting plane defects in any direction of the material. which was this study's basis for choosing Ultrasonic Testing in combination with machine learning.

In 2008, Carvalho [3] used radiography, manual detection, and automatic acoustic technology to classify three types of industrial piping defects, namely lack of penetration, lack of fusion, and undercut. Among them, the radiographic inspection technology adopted γ-ray and X-ray, while automatic scanning was performed by an inspection vehicle with magnetic wheels. By using scanned data, the defect size was estimated by detecting the discontinuity of the weld combining MATLAB. The results showed that Ultrasonic Testing (UT) technology had obvious advantages

over other technologies. The Carvalho group attempted to utilize Artificial Neural Network (ANN) method to classify defects. After the ultrasonic signal was preprocessed and smoothed, it was used as a featured input. Results showed that Artificial Neural Network (ANN) cannot classify different types of defects but could examine the existence of defects.

In 2009, Caiping Zhao [4] developed a diagnostic system for detecting defects using pipeline ultrasonic guided wave test data. This system selected the eigenvalues such as the amplitude of the reflected signal to demonstrate the features and defects of the pipeline structure in the detection diagram. Fifteen classes of defects were classified by the Back Propagation (BP) algorithm, and the recognition rate of elbows and welding joints were the highest and reached more than 90%, which proved that the recognition rate of the neural network was high with stable results.

In 2019, Roberto [5] used Ultrasonic Testing (UT) flaw tracker to collect multiple sets of ultrasonic data including the length, depth, and location of cracks. After professionals classified and trained the data using machine learning, resulted in pinpoint accuracy of Support Vector Machine (SVM) to classify various classes of defects.

In 2019, Tripathi [6] classified microdamage using a piezoelectric ceramics sample. In general, it is difficult for experts to classify the damaged value in material with smaller than 100 μm cracks in depth. Through machine learning and deep learning, the K-Nearest Neighbor (KNN) of machine learning was very suitable to classify micro-damaged ceramic plates in the counterpart of Convolutional Neural Networks (CNN) deep learning. Therefore, the deep learning techniques cannot offer specific advantages over simple machine learning algorithms. In terms of data feature selection, the features of frequency domain were found better than that in time domain.

In 2020, A. Mardanshahi [7] proposed a new model to automatically detect and classify the crack density of composite materials using guided wave propagation and artificial intelligence. They used Non-Destructive Testing (NDT) program via the antisymmetric Lamb wave to test samples with different crack densities, and then extracted information such as the amplitude and wave speed of the Lamb wave from the collected signals. After training the set, the classification accuracy of the Support Vector Machine (SVM) showed the highest performance of 91.7%, while the classification accuracy of the Neural Network (NN) reached a maximum of 88%.

1.1.2 Applications in Bioengineering

Wort is the liquid extracted from the mashing process during the brewing of liquor. In 2019, Fan Zhang [8] determined the wort production quality of beer by combining Non-Destructive Testing (NDT) technology with machine learning. Since the production data was high-dimensional, a dimensionality reduction method to obtain a concise latent space was adapted, which was further used for data analysis to control the wort production quality. In this experiment, Near-Infrared Spectroscopy (NIR) technology was used primarily to collect production data, and the obtained data was combined with machine learning methods such as Principal Component Analysis (PCA) to analyses wort production quality. This group successfully demonstrated the use of low-dimensional data to represent high-dimensional data.

In 2020, Te Ma [9] used near-infrared hyperspectral imaging combined with deep learning to predict seed viability. The experimental data were related to the internal molecular vibration information (chemical composition difference) and spatial distribution of seeds. The Principal Component Analysis (PCA) method and Support Vector Machine (SVM) method were used for training. The results showed that even the naturally aged seed test set could produce about 90% accuracy in classification as compared to the normal seed test set with a classification accuracy of nearly 95%, which proved the reliability of these two methods for predicting seed viability.

1.1.3 Applications in Medical Science

In 2017, Burlina [10] used ultrasound imaging combined with machine learning and deep learning for the diagnosis of muscle inflammation. Eighty subjects in this experiment were divided into three groups of patients with different muscle inflammation and one group of healthy individuals was considered for reference. In terms of machine learning methods, the echo intensity of muscle and fat was used as the characteristics for training by using the random forest method. Meanwhile, Deep Convolution Neural Network (DCNN) used to train ultrasound images. The results indicated that predictions performance made by Deep Convolution Neural Network (DCNN) method was higher than machine learning method random forests.

In 2020, Zhengsi Xiong [11] performed an Non-Destructive Testing (NDT) of liver cancer with a combination of machine learning methods. The experiment was designed to collect breath data from healthy subjects and liver cancer patients, and dimensionality reduction processing on the generated data was performed. Data such as sensor temperature and relative humidity were selected as features, and then machine learning, the Support Vector Machine (SVM) algorithm, was applied to classify the data. The Support Vector Machine (SVM) algorithm with a linear kernel function showed the best classification effect.

1.1.4 Applications in Physics

In 2018, William Sorteberg [12] research group created a data set to simulate wave motion using the Long Short-Term Memory (LSTM) method in order to build a predictive deep neural network with three main modules. After testing the test set, the structural similarity index decreased during longer-term predictions, and the neural network can predict the future information up to 80-time steps only.

In 2019, Rautela M [13] used high-frequency tone-burst signals as the excitation waveguide in the experiment and the time domain as the spatial feature input in the deep learning framework. They used Convolution Neuron Network (CNN), Recurrent Neural Network (RNN), and Long Short-Term Memory (LSTM) to detect the signal to simulate cracks in the waveguide. This article supports the evidence that a deep learning framework can provide perfect binary classification. The significance of this study highlights the fact that deep learning algorithms are promising tool for learning guided wave data sets.

In 2019, Yohei [14] combined Convolution Neuron Network (CNN) with sensors to create a new wave-front sensor. The principle was that the image receiver used the deep learning method to estimate directly the Zernike coefficient by preprocessing the measured value of the intensity of a single light source, indicated that data preprocessing can improve the accuracy of the wavefront prediction through deep learning. Furthermore, Zernike polynomials were used to describe the properties of the wavefront.

1.1.5 Applications in Other Domain

In 2017, Pingping Zhu [15] used deep learning to recognize and classify targets in underwater sonar images and utilized Convolution Neuron Network (CNN) to extract image features, and then a Support Vector Machine (SVM) was used for classification after extraction. They confirmed that the combination of Support Vector Machine (SVM) with Convolution Neuron Network (CNN) showed the best effect on the classification of underwater sonar images as compared to the combination of Support Vector Machine (SVM) with local binary pattern and Support Vector Machine (SVM) with the histogram of oriented gradients.

In 2018, Tomasz [16] developed a hybrid Computed Tomography (CT) scanner with special sensors for designing humidity analysis. The scanner was used to generate data for training, which combined with the neural network method to create the wall humidity images, leading to reflect the humidity inside the wall. They demonstrated that the estimation of wall humidity in combination with neural networks is better than traditional least angle regression or ElasticNet methods.

1.2 Outline of the Thesis

(1) Generate datasets using Fortran program based on the theory of elasticity

(2) Use Machine Learning and Deep Learning to predict crack information.

- Machine Learning Algorithms:
    - Support Vector Machine (SVM)
    - K-Nearest Neighbors (KNN)
    - Random Forest (RF)
    - Extremely Randomized Tree (ERF)
- Deep Learning Algorithms:
    - Simple Recurrent Neural Network (SRNN)
    - Long-Short Term Memory (LSTM)
    - Gated Recurrent Unit (GRU)

(3) Introduce different data types and compare the training results

(4) Create a table of results to reflect the predictive performance between different method.

(5) Compare the prediction performance of Machine Learning and Deep Learning.

(6) Analysis and discuss the output results.

1.3 Contributions

Firstly, the current study identified a suitable algorithm for predicting crack defects in pipelines, that is because many conclusions on the ideal prediction method is not uniform, to discover an algorithm that is optimal for pipeline crack prediction was the goal of this study.

Secondly, the wave response coefficients are used as features to predict pipelines' cracks. The coefficients be got are differ from the features about wave velocity or wave form amplitude, that is because the wave response coefficients are frequency domain signals which transferred form time domain signals by Fourier transform.

Thirdly, the using of the suggested data type and features of prediction cracks is reliable and accurate. Because varies data sets can be departed into different data types, and each data set has different prediction performance, that is also the same as features. Thus, to improve the prediction accuracy, finding an optimal data type and feature are needed.

Chapter II Methodology

2.1 The Fundamentals of Waves in a Cylinder

This chapter will briefly cover wave propagation in a cylinder using the formula and theory described by Datta and Shah [17]. First, the fundamental equations of wave propagation in a cylinder are reviewed. After obtaining the wave solution via the semi-finite element approach, the wave solution is used to solve the wave reflection and transmission problems.

2.1.1 Equations of Motion

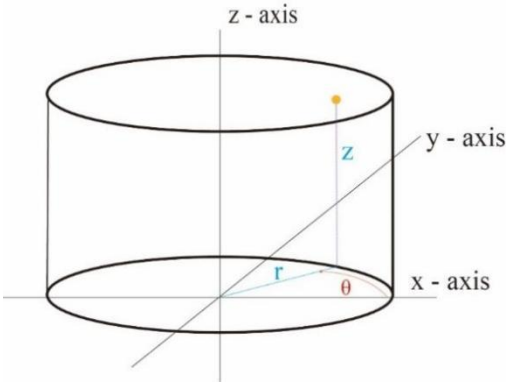

Fig. 1 Cylindrical Coordinates

Cylindrical coordinates are used to solve the wave propagation problem with each point of three displacement components:

① u_r displacement component in the radial direction.

② u_θ displacement component in the circumferential direction.

③ u_z displacement component along the vertical axis.

The equations of motion expressed in terms of stresses are:

$$\frac{\partial \sigma_{rr}}{\partial r} + \frac{1}{r}\frac{\partial \sigma_{r\theta}}{\partial \theta} + \frac{\partial \sigma_{rz}}{\partial z} + \frac{1}{r}(\sigma_{rr} - \sigma_{\theta\theta}) = \rho \ddot{u}_r \tag{2.1}$$

$$\frac{\partial \sigma_{r\theta}}{\partial r} + \frac{1}{r}\frac{\partial \sigma_{\theta\theta}}{\partial \theta} + \frac{\partial \sigma_{\theta z}}{\partial z} + \frac{2}{r}\sigma_{r\theta} = \rho \ddot{u}_{\theta} \tag{2.2}$$

$$\frac{\partial \sigma_{rz}}{\partial r} + \frac{1}{r}\frac{\partial \sigma_{\theta z}}{\partial \theta} + \frac{\partial \sigma_{zz}}{\partial z} + \frac{1}{r}\sigma_{rz} = \rho \ddot{u}_{z} \tag{2.3}$$

Here σ_{ij} is the stress components, i, j = r, θ, z. ρ is mass density. Double dots represent the partial derivative with respect to time. The relationship between strain and displacement is given by:

$$e = (L_r + L_\theta + L_z)u \tag{2.4}$$

Here, $u = (u_r, u_\theta, u_z)^T$ is displacement vector and L_r, L_θ, and L_z represent the partial derivatives of r, θ, and z, respectively, and are given by:

$$L_r = \begin{bmatrix} \frac{\partial}{\partial r} & 0 & 0 \\ \frac{1}{r} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \frac{\partial}{\partial r} \\ 0 & \frac{\partial}{\partial r} - \frac{1}{r} & 0 \end{bmatrix} \quad L_\theta = \begin{bmatrix} 0 & 0 & 0 \\ 0 & \frac{1}{r}\frac{\partial}{\partial \theta} & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \frac{1}{r}\frac{\partial}{\partial \theta} \\ 0 & 0 & 0 \\ \frac{1}{r}\frac{\partial}{\partial \theta} & 0 & 0 \end{bmatrix}, \quad L_z = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \frac{\partial}{\partial z} \\ 0 & \frac{\partial}{\partial z} & 0 \\ \frac{\partial}{\partial z} & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{2.5}$$

According to Hooke's law, the relation expression is listed as follow:

$$\begin{Bmatrix} \sigma_{rr} \\ \sigma_{\theta\theta} \\ \sigma_{zz} \\ \sigma_{\theta z} \\ \sigma_{zr} \\ \sigma_{r\theta} \end{Bmatrix} = \begin{bmatrix} c_{11} & c_{12} & c_{13} & c_{14} & c_{15} & c_{16} \\ & c_{22} & c_{23} & c_{24} & c_{25} & c_{26} \\ & & c_{33} & c_{34} & c_{35} & c_{36} \\ & & & c_{44} & c_{45} & c_{46} \\ & \text{sym} & & & c_{55} & c_{56} \\ & & & & & c_{66} \end{bmatrix} \begin{Bmatrix} e_{rr} \\ e_{\theta\theta} \\ e_{zz} \\ \gamma_{\theta z} \\ \gamma_{zr} \\ \gamma_{r\theta} \end{Bmatrix} \tag{2.6}$$

Here, e_{ii} is the normal strain components, i = r, θ, z. and γ_{ij} is the engineering shear strain component. i, j = r, θ, z, i ≠ j. The vector on the left side of the equation represents six distinct stress components, whereas six distinct strain components are represented in the vector on the right.

The boundary conditions are that the inner and outer surfaces are traction free.

2.1.2 Semi Analytical Finite Element

As shown in Figure 2, the section of pipeline is divided into N parts. The black circles are one of the finite elements, R_k is the inner diameter of the ring, R_{k+1} refers to the outer diameter of the ring, h_k stands for the thickness of the ring, H is the thickness of the entire pipeline, k^{th} sublayer refers to the k^{th} ring.

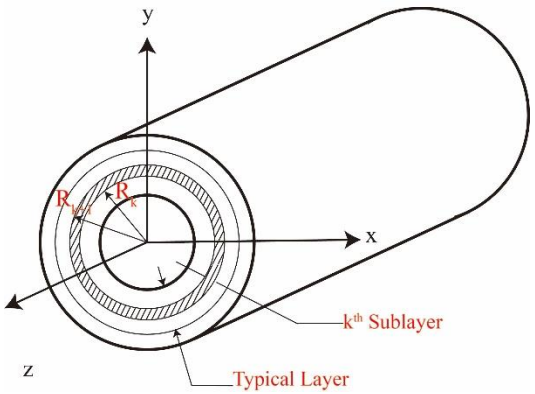

Fig. 2 Semi Analytical Finite Element in Cylinder

Divide the composite cylinder into several coaxial cylinders and used a quadratic polynomial interpolation function to represent the displacement distribution on the thickness of the sublayer in the radial direction. In the k^{th} sublayer, the displacement component of a certain point is as follows:

$$\{U\} = [N(r)]\{q\} \tag{2.7}$$

where,

$$\{U\} = \langle \tilde{u}\ \tilde{v}\ \tilde{w} \rangle^T \tag{2.8}$$

$$\{q\} = <\tilde{u}_k^b\ \tilde{v}_k^b\ \tilde{w}_k^b\ \tilde{u}_k^m\ \tilde{v}_k^m\ \tilde{w}_k^m\ \tilde{u}_k^f\ \tilde{v}_k^f\ \tilde{w}_k^f>^T$$

$\{q\}$ represents the displacement of three nodes in a unit, each with three component vectors, for a total of 9 vectors in $\{q\}$. Here b, m, and f describe the first, middle, and last points, respectively.

$\tilde{u}$, $\tilde{v}$, $\tilde{w}$ indicate the three displacement components, r indicates radius. [N(r)] represents the function of the interpolation matrix. The interpolation polynomials n_i (i=1,2,3) are quadratic functions of the radial variable defined as:

$$n_1 = 1 - 3\eta + 2\eta^2 \tag{2.9}$$

$$n_2 = 4\eta - 4\eta^2$$

$$n_3 = -\eta + 2\eta^2$$

Where $\eta = \frac{(r-r_k)}{h_k}$, h_k being the thickness of the sublayer, and r_k being the radial coordinate of the inner surface of the k^{th} sublayer.

The final finite element equation has the form:

$$(-k^2[K_1] - ik[K_2] - [K_3] + \omega^2[M])\{Q_0\} = 0 \tag{2.10}$$

Here, K_1, K_2, and K_3 are all stiffness matrices; M is the mass matrix; the node displacement vector Q has the wave form solutio0n as:

$$\{Q\} = \{Q_0\}e^{i(m\theta+kz-\omega t)} \tag{2.11}$$

Here, $\{Q_0\}$is the amplitude of the wave solution; m is circumferential wave number, m= 0, ±1, ±2 …; k is axial wave number.

The matrices [M], $[K_1]$, $[K_2]$, and $[K_3]$ are defined below.

$$[M] = \int_0^H \rho\,[N]^T[N]rdr \tag{2.12}$$

$$[K_1] = \int_0^H [b]^T\,[C][b]rdr$$

$$[K_2] = \int_0^H [b]^T\,[C][a] - [\bar{a}][C][b]rdr$$

$$[K_2] = \int_0^H [a]^T\,[C][a]rdr$$

The nonzero elements of the 6 × 9 matrix [a] are as follows:

$$a(1,1) = \frac{dn_1}{dr}, a(1,4) = \frac{dn_2}{dr}, a(1,7) = \frac{dn_3}{dr} \tag{2.13}$$

$$a(2,1) = \frac{n_1}{r}, a(2,2) = im\frac{n_1}{r}, a(2,4) = \frac{n_2}{r}$$

$$a(2,5) = im\frac{n_2}{r}, a(2,7) = \frac{n_3}{r}, a(2,8) = im\frac{n_3}{r}$$

$$a(4,3) = im\frac{n_1}{r}, a(4,6) = im\frac{n_2}{r}, a(4,9) = im\frac{n_3}{r}$$

$$a(5,3) = \frac{dn_1}{dr}, a(5,6) = \frac{dn_2}{dr}, a(5,9) = \frac{dn_3}{dr}$$

$$a(6,1) = im\frac{n_1}{r}, a(6,2) = \frac{dn_1}{dr} - \frac{n_1}{r}, a(6,4) = im\frac{n_2}{r}$$

$$a(6,5) = \frac{dn_2}{dr} - \frac{n_2}{r}, a(6,7) = im\frac{n_3}{r}, a(6,8) = \frac{dn_3}{dr} - \frac{n_3}{r}$$

The nonzero elements of [b] are:

$$b(3,3) = b(4,2) = b(5,1) = n_1 \tag{2.14}$$

$$b(3,6) = b(4,5) = b(5,4) = n_2$$

$$b(3,9) = b(4,8) = b(5,7) = n_3$$

It is noted that $[K_1]$ and [M] are real and symmetric, $[K_2]$ is skew-Hermitian, and $[K_3]$ is Hermitian.

The solutions of equation (2.10) give the wave number and the corresponding wave modes. For a given incident wave, the wave coefficients of the response will be determined via superposition of wave modes. The method is depicted in Figure 3 below:

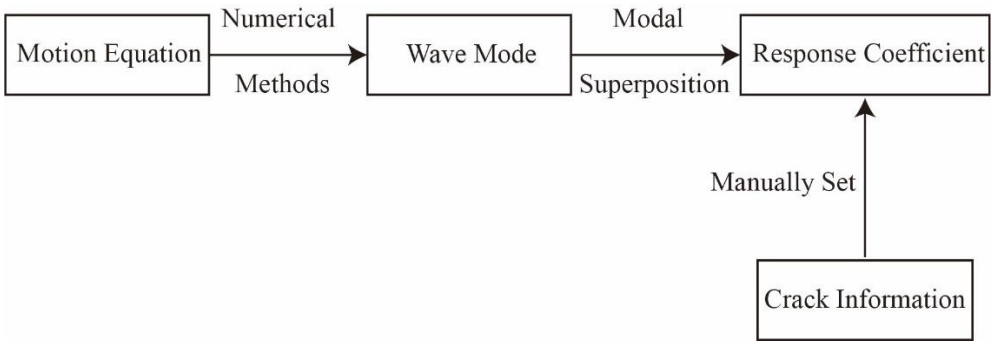

Fig. 3 Fortran Process

2.2 Selection of Data Characteristics

The reported studies used structural response data (for example, velocity and acceleration) collected from experiments as the training data to train the neural networks in pipeline nondestructive testing; however, to the best knowledge of the author, there are no published literatures for directly using wave coefficients in the study. Additionally, the amount of experimental data is limited, which significantly impacts the classification results' correctness. The distinction between this study and others is that this study use wave response coefficients as the primary features of the data, which are used for training in machine learning and deep learning, and making a prediction on the studied pipeline's crack parameters. After training the model, the fracture information may be predicted using the input measurement's wave response coefficient.

Here is a summarization of the features will be used in the study:

- Transmission Coefficient
- Reflection Coefficient
- Circular Frequency
- Circumferential Wave Number
- Thickness Over Mean Radius
- Materials

2.3 Combination of Machine Learning

Machine learning is a subfield of artificial intelligence and computer science that focuses on replicating how humans learn and steadily improving their accuracy via data and algorithms. It is not a computer-specific algorithm but a collective name for various algorithms, of which Deep Learning is one. Machine Learning's fundamental strategy is to solve real-world problems by abstracting them into mathematical models and utilizing machines to solve these mathematical problems, ultimately resolving the real-world problem. In this study's example, by utilizing a Machine Learning method, this study extracted information about pipeline breaks from a variety of datasets.

2.3.1 Determination of Machine Learning Algorithm

Since the current study involves anticipating pipeline fracture information, a machine learning technique with a regression model was required. According to Nerseen's analysis of machine learning models for time series prediction [18], the machine learning methods for regression models are Support Vector Machine (SVM), Random Forest (RF), Extremely Randomized Tree (ERT), and K-Nearest Neighbors (KNN).

2.3.2 Principles of Machine Learning Algorithms

2.3.2.1 Support Vector Machine (SVM) Algorithm

Support Vector Machine (SVM) can be classified into two types: the Support Vector Classification (SVC) algorithm, which is appropriate for classification issues and datasets, and the Support Vector Regression (SVR) algorithm, which is appropriate for regression problems and datasets (shown in Figure 4). SVC denotes the capacity to maximize the distance between the nearest sample points in the hyperplane; SVR, on the other hand, denotes the ability to minimize the distance between the farthest sample points in the hyperplane. In this experiment, the Support Vector Regression is employed, the red and blue points represent the data set used.

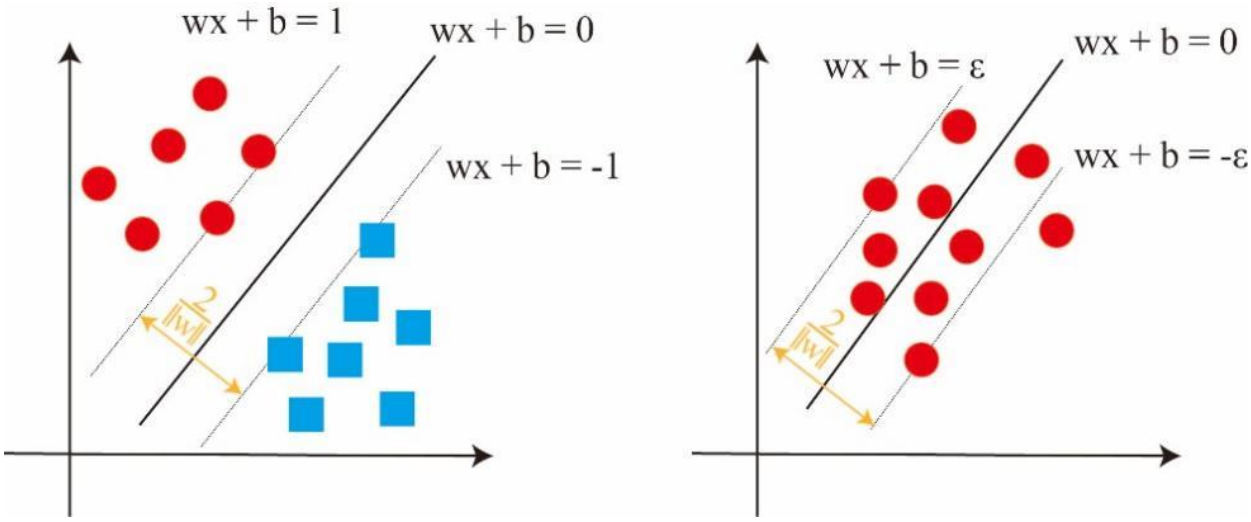

Fig. 4 Support Vector Classifier (left) and Support Vector Regression (right)

Here, $w \cdot x + b = \pm 1$ and $w \cdot x + b = \pm \varepsilon$ are boundary lines on both sides of SVC and SVR, respectively. $w \cdot x + b = 0$ represents the hyperplane. $\frac{2}{|w|}$ is the distance between the two dashed lines. w is the normal vector which decide the direction of hyperplane, and b is the intercept decide the distance between hyperplane and the origin. This study are using (w, b) to represent this hyperplane. ε is the element of relaxation. The greater the value of ε, the more closely the sample point approaches the hyperplane.

The prediction function of SVR is to make the loss of all data points within the margin boundary equals to 0, and the points outside the margin boundary are the support vectors of SVR. All this study needs to do is to ignore the points within the boundary lines and regress the remaining points. Due to the high dimension of this study's data sets, utilizing the kernel function to transfer these samples to a higher dimension before doing regression is needed.

2.3.2.2 Random Forest Algorithm

Random Forest (RF) is built of several decision trees. Besides, there is no connection between the various decision trees. Random Forest (RF) is a kind of random sampling or random

selection of features that may help avoid overfitting. When this study feed data sets into Random Forest (RF), each decision tree is predicting independently. Each decision tree will give a prediction value. The final prediction result is calculated as the mean of all those decision tree's predictions.

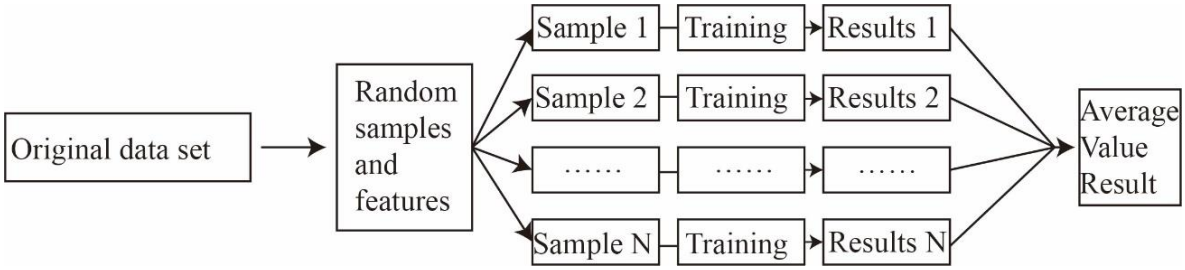

Fig. 5 Random Forest Structure Diagram

2.3.2.3 Extremely Randomized Tree Algorithm

In contrast to the conventional Random Forest (RF), which selects the best segmentation, an Extremely Randomized Tree (ERF) is a version of the Random Forest (RF) algorithm that randomly selects the segmentation. This results in a higher degree of generalization and a shorter calculation time when compared to Random Forest (RF). As a result, Extremely Randomized Tree (ERF) frequently outperforms Random Forest (RF) in prediction accuracy.

2.3.2.4 K-Nearest Neighbors Algorithm

The value of the anticipated point is determined in Figure 6 by averaging the values of the K points nearest to it. In this case, the "closest distance" could be the Euclidean or another distance. For instance, suppose K equals three, and the point value in this study wish to predict is dependent on the three nearest red points. Each red point in the graphic corresponds to a particular set of data sets this study used, while the green dots correspond to the anticipated values. When the KNN solves a regression model problem, the average algorithm is typically employed, which means that the regression prediction value is calculated using the average value of the sample output from the nearest K samples.

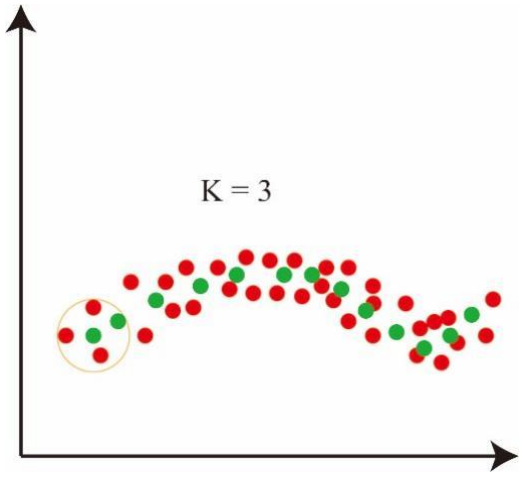

Fig. 6 K-Nearest Neighbors

2.3.3 Selection of Hyperparameters in Machine Learning Code

Numerous parameters must be modified in machine learning programs. These parameters affect the algorithm's prediction performance. The optimal parameter combination for the relevant algorithm can be determined through repeated training datasets.

2.3.3.1 Support Vector Machines (SVM)

The Support Vector Machine (SVM) code mainly uses parameters C and gamma in this experiment:

- C: the penalty coefficient; the higher the value of C is, the easier it is to overfit the data. Conversely, underfitting can occur with smaller values of C.
- Gamma: a default parameter when the RBF function is chosen as the kernel. After mapping to the new feature space, it implicitly defines the data distribution. The greater the gamma value, the fewer support vectors there are; the smaller the gamma value, the more support vectors there are. The number of support vectors has an effect on the training and prediction speeds.

2.3.3.2 K-Nearest Neighbors (KNN)

There are three commonly used hyperparameters for KNN, which are as follows:

- K: indicates the number of 'neighbors,' the default value is three, which means sending the three closest samples. In brief, a low K number implies that the entire model is complex and prone to overfitting; on the other side, a high K value shows that the error is large and the prediction accuracy is low.
- Weight: Mainly used to return results. The default setting is uniform.
- p: indicates the variable in Minkowski distance formula. Minkowski Distance is a generalization of Euclidean distance, which is a general expression of multiple distance measurement formulas. When p = 2, this study obtain the Euclidean distance.

2.3.3.3 Random Forest & Extremely Randomized Tree Algorithm

Random Forest and Extremely Randomized Tree parameters need to be adjusted to ensure consistency. The parameters that must be altered are divided into two sections. The first section contains parameters for the Bagging framework, while the second section contains the CART Decision Tree parameters.

Partial parameters of Bagging framework:

- n_estimators: indicates the maximum number of iterations of the weak learner. If the value of N_ESTIMators is small, underfitting is likely to occur. Conversely, if the n_ESTIMators value is high and easy to overfit, the default value is 100.
- oob_score: indicates whether to choose to use out-of-bag samples to evaluate the quality of the model. Its default value is False.

Partial parameters of the CART Decision Tree:

- max_features: represents the maximum number of features in the random forest after partitioning. The default option is "None," which implies that all feature numbers are

evaluated when dividing. The total number of features to consider is indicated by specifying this feature as an integer. The value of max features must be increased because the wave response coefficient generates a significant number of reflected and transmitted waves (more than 50).

- min_samples_split: specifies the minimal number of samples required to subdivide internal nodes. This parameter limits the conditions under which the subtree may continue to be divided, and the default value is 2.

2.4 Combination of Deep Learning

2.4.1 Determination of Deep Learning Algorithm

Deep learning emerged as a technique for optimizing artificial neural networks using backpropagation. It was first primarily represented by a Multilayer Perceptron. However, due to the issue of vanishing or exploding gradient during the training phase, no significant advances in neural network research have been made. Deep learning has advanced significantly in recent years, owing to the availability and utilization of large amounts of data and the rapid increase in the computational power of computers. In comparison to Convolutional Neural Networks (CNN), Artificial Neural Networks (ANN), and other methods, Recurrent Neural Networks (RNN) are better at dealing with time series problems and emphasizing sequence order, whereas CNN is better at processing spatial windows. ANNs are a collection of multi-layer neurons, usually referred to as feedforward neural networks, that are primarily used to solve tabular, text, and picture data problems. A cyclic neural network is generated to manage sequential processing jobs such as time-series data, literal expressions, etc. An RNN's structure includes a memory function, which enables it to analyze sequential input with dependencies and has demonstrated exceptional performance in various natural language processing applications. In 2018, I. Jahan and S. Z. Sajal [19] forecasted the stock price using RNN. Long Short-Term Memory (LSTM) is an RNN version that outperforms the normal RNN in many applications. F. Altch é [20] employed LSTM in 2017 to forecast the trajectory of highway traffic. The LSTM algorithm avoids the vanishing gradient and has a more significant memory, but the recirculating network is effectively. The

Gated Recurrent Unit (GRU) is a newer generation of RNN, and it is also pretty similar to LSTM, because of GRU has less operations, so that it is little speedier than LSTM.

Determining Simple Recurrent Neuron Network (RNN), Long-Short Term Memory (LSTM), and Gaited Recurrent Unit (GRU) as deep learning algorithms.

2.4.2 Principles of Deep Learning Algorithms

2.4.2.1 Recurrent Neural Network (RNN) Algorithm Principles

Human reading habits are comparable to recurrent neural networks (RNN). The RNN accumulates information after reading using the state vector h_t, just like the brain does each time a human reads a word. It is distinct from the typical neural network model in that it has recurrent connection between previous output to current input. Figure 7 illustrates the structure of an RNN.

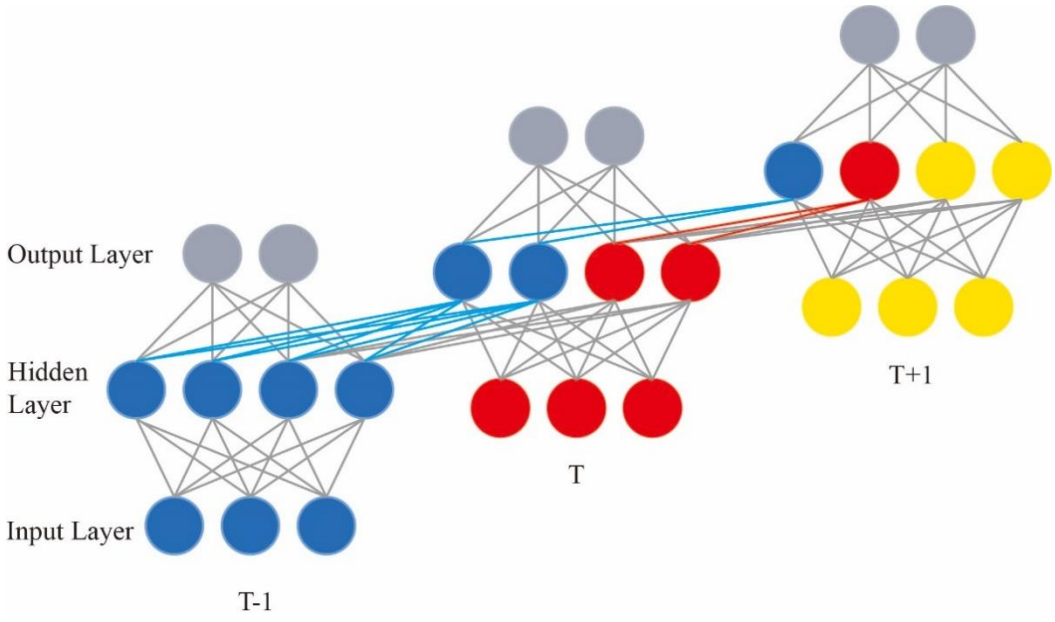

Fig. 7 Example of Recurrent Neural Network (RNN)

Here, the figures' circles indicate neurons, while the various colors reflect various time. As can be seen, the RNN's hidden layer at time T contains information from the preceding time T-1.

The primary distinction between RNN and other algorithms such as Artificial Neural Network (ANN) or Convolutional Neural Network (CNN) is that the weight connections between the neurons in the layers are established in such a way that the output at each moment is related to the current input and the previous output. The following diagram illustrates structure of RNN:

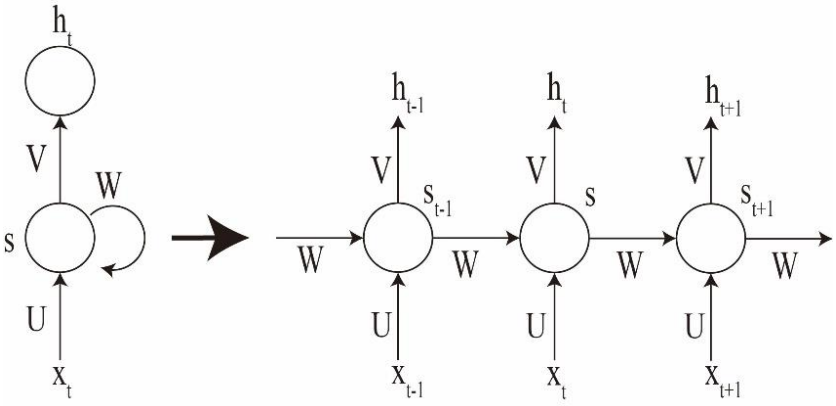

Fig. 8 Basic structure of a standard recurrent neural network (RNN)

Data will be preprocessed into vector x_t first, then input into the hidden layer S to update the state vector through the parameter matrix W, and finally output h_t to store the state information. In this process, the parameter matrix W of the whole RNN chain remains constant. The hidden layer state S_t of the current time node is related to the current time t input x_t and the hidden layer state S_{t-1} of the previous time.

$$s_t = \varphi(Wx_t + Us_t - 1) \tag{2.11}$$

Here U and V are weight matrices, φ is a logical S-shaped function or hyperbolic tangent function, and W is the circulant weight matrix of state transition.

It is worth noting that Figure 8 does not mean that the RNN has only three neural networks, but it means that the same neural network is used three times at three different time points.

2.4.2.2 Principle of Long Short-Term Memory (LSTM)

The long short-term memory (LSTM) model was developed by Hochreiter and Schmidhuber [21]. It is a variant of the recurrent neural network. Proposed in 1997, the LSTM may be used to analyze data with distant nodes in a time series and efficiently capture information about significant time nodes in previous time series to make more precise inferences about the current moment's content. Compared to the more straightforward cyclic design, which is more reliable for long-term learning, the LSTM has demonstrated superior performance in Automatic Speech Recognition (ASR), machine translation, image description creation, and other applications. LSTM solves problems with the disappearing gradient and exploding gradient of standard RNNs in various tasks. LSTM networks converge more easily than RNN networks, which has allowed them to gradually supplant RNN as the favored model for sequential task processing. Figure 9 illustrates its interior structure.

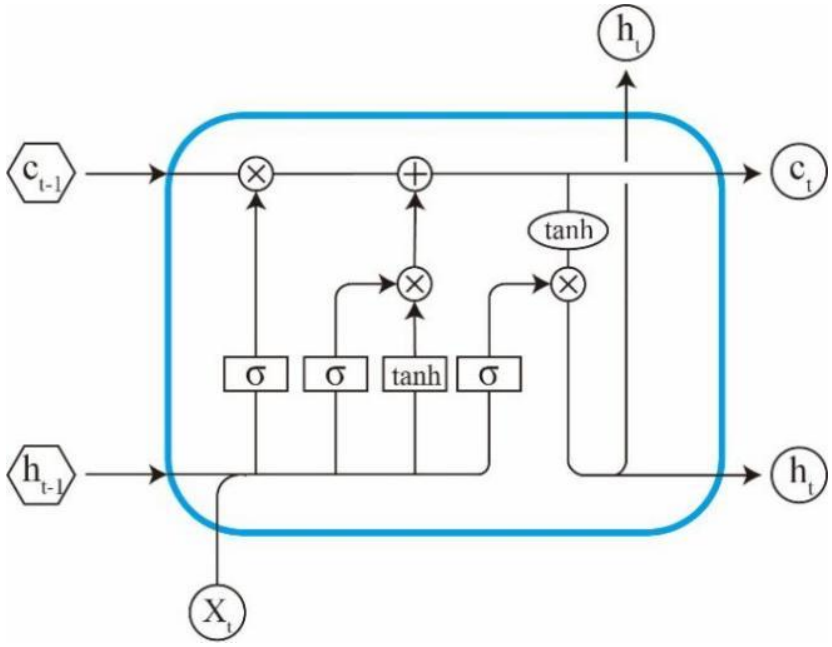

Fig. 9 Principle of Long-Short Term Memory (LSTM)

The fundamental concept of LSTM is to interact with the Cell State via "three gates" and modify the information held by the Cell, which are the Input Gate, Forget Gate, and Output Gate. A Cell State is analogous to a conveyor belt that runs parallel to the chain, with very tiny interactions

that allow information to flow freely. Additionally, LSTM can add or delete the Cell States, which is regulated by a gate structure and is a mechanism to allow information to pass through selectively. They are composed of a layer of Sigmoid neural networks and a multiplication operation at the element level. The output values of the sigmoid layer determine whether the corresponding portion of the information is passed. Three gates protect and govern the Cell State in an LSTM. Thus, the LSTM has four inputs that ultimately result in a single output. Its internal structural diagram, depicted in Figure 10, can be simplified.

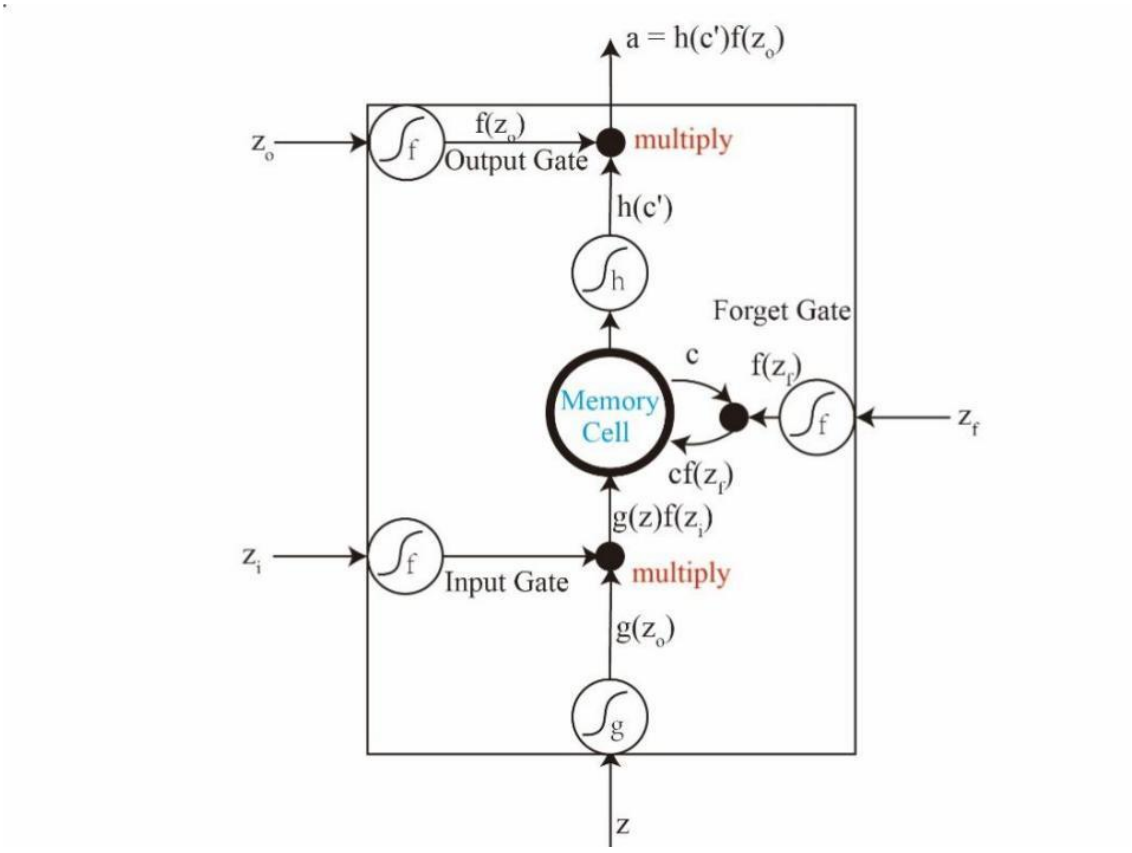

Fig. 10 Inner structure of a Long Short-Term Memory (LSTM)

The black dot denotes the elementwise multiplication of two vectors, while f denotes the sigmoid activation function, which regulates the value between 0 and 1. Its value indicates the state of the gate's opening. By definition, when the value is 0, the gate is closed. Notably, the Hyperbolic Tangent Function is used as the activation function in g and h. Z is an external input, and the

inputs of the three gates are represented by Z_i, Z_f, and Z_o. The c in the middle is short for the memory cell and the output is represented by a. The whole process is input Z and Zi, which are transformed into $f(z_i)$ and $g(z)$ through the activation function, and $g(z)f(z_i)$ is obtained after elementwise multiplication. Meanwhile, $f(z_f)$ is obtained through the activation function, and $cf(z_f)$ is obtained by multiplying the c value previously stored by $f(z_f)$. By addition, the updated memory cell value c' can be expressed as follows:

$$c' = g(z)f(z_i) + cf(z_f) \tag{2.15}$$

$h(c')$ is then obtained through activation function h. It is then multiplied by $f(z_o)$, generated by the output gate to obtain the output a. The formula of a is as follows:

$$a = h(c')f(z_o) \tag{2.16}$$

In neural networks, a neuron is represented by an LSTM. In general, multiple LSTMS are used. Only two LSTMS are shown here in Figure 11.

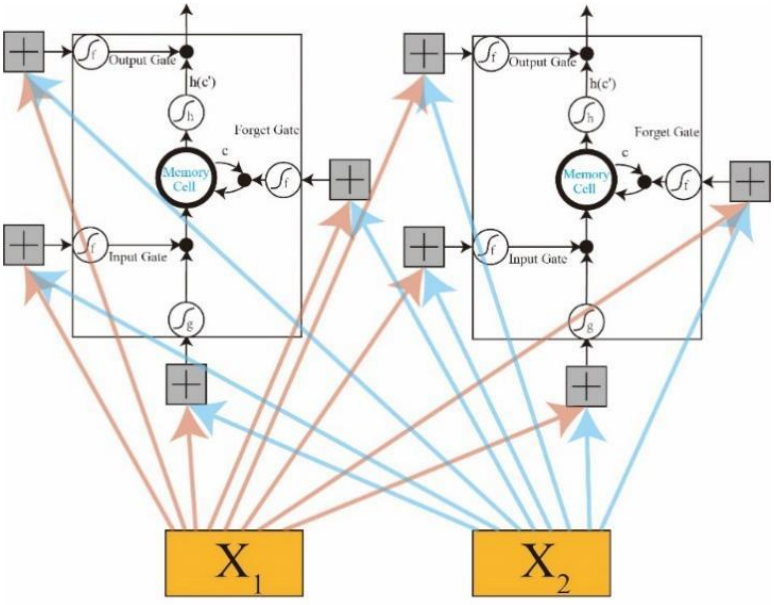

Fig. 11 Neural Network Figure of Long-Short Term Memory (LSTM)

2.4.2.3 Principle of Gated Recurrent Unit (GRU)

In fact, the Gated Recurrent Unit (GRU) model is a variant of the LSTM model. Figure 12 shows the GRU hidden layer cell.

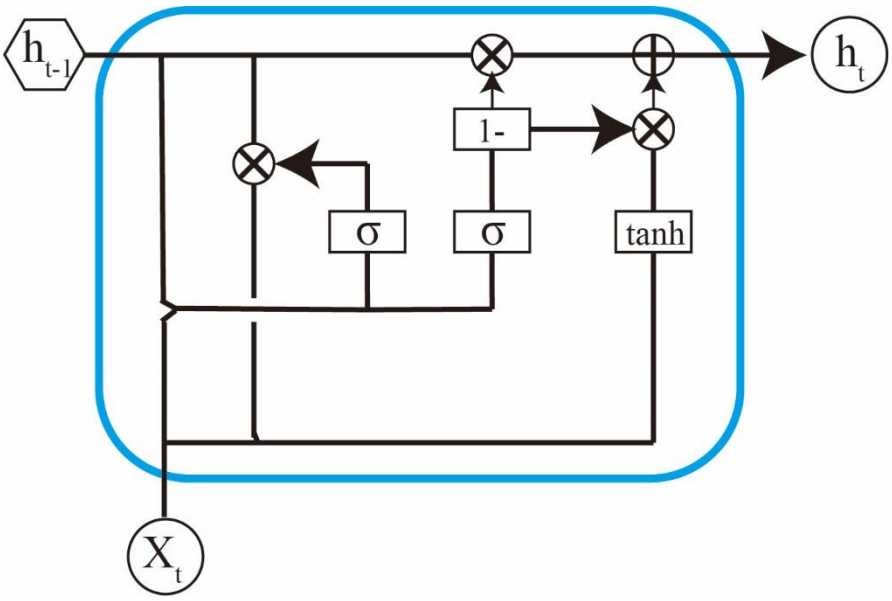

Fig. 12 GRU cell

According to Jeffrey [22], here, h_{t-1} is previous output, x_t and h_t is current input and output. '×', '+' and '-1' are logical operator, means multiply, plus and minus one.

Compared to the LSTM, the structure of GRU doesn't has cell state, which means GRU has less operations. GRU has 2 gates here, a reset gate and an update gate. The update gate acts similar to forget gate and input gate of LSTM, it decides what information to throw away and what new information to add. The reset gate is a gate used to decide how much pass information to forget.

2.5 Selection of Hyperparameters for Deep Learning Code

Although LSTM is a version of RNN and GRU is a derivative of LSTM, they all use the same hyperparameters. The following table summarizes the information required for this experiment's hyper-parameters:

- num_epochs: The number of epochs. An epoch is the number of times that all data is trained once, and the number represented by epoch is the total number of training rounds. The length of the epoch is proportional to the diversity of the dataset. The more diversification there is, the longer the epoch should be.
- Batch_size: indicates how frequently a subset of the data is supplied to the network for training. The optimal batch size range is mostly determined by the convergence rate and stochastic gradient noise.
- INIT_LR: represents the Learning Rate, a hyperparameter used to update the weight throughout the gradient descent process. The learning rate's magnitude dictates whether and when the objective function can converge to the local minimum.
- num_ Layer: specifies the size of the RNN's hidden layer. By selecting an appropriate hidden layer, problems such as gradient explosion can be avoided. By selecting a suitable hidden layer, gradient explosion and other issues can be avoided.
- hidden_ Size: indicates the number of neurons in each hidden layer of the RNN.

Chapter III Process and Results

3.1 Pre-Setting of Hyperparameters and Pipeline Parameters

Hyperparameters control the solving rate, the solution's reliability, and the optimization problem is the learning effect. A fine hyperparameter can assist in rapidly identifying the optimal solution to the minimization problem and force the model to match the data better through effective generalization.

3.1.1 Settings of Pipeline Parameters

When the pipeline is divided into 40 layers, the results converge. The pipeline is separated into 40 levels to calculate the number of flaws. As a result, there are 81 points on the section line along the pipeline's radius. According to formula 2.7, this study must first provide the appropriate point's interpolation [N(r)], after which the point's displacement component value can be determined.

The majority of Pipeline Material (PM) quantities are set to one. Among them are five widely used isotropic materials: iron, copper, magnesium, titanium, and aluminum, in addition to an anisotropic composite.

Thickness Over Mean Radius (TOMD) is typically between 0 and 2. On the other hand, a low TOMR suggests that the pipeline is extremely thin; a solid cylinder has an TOMR of 2. In this experiment, TOMR values were set between 0.1 and 0.4 to avoid exceeding 0.4, which would have resulted in a loss of application value in real-world engineering projects.

The circular Frequency of the Pipeline (CFP) is the incident wave's frequency. In general, the greater the frequency, the more waves propagate, requiring more calculation. On the other hand, the lower the frequency, the fewer waves may be received. Additionally, when the frequency exceeds a particular threshold, it results in a cliff decline in phase velocity, detrimental to data analysis. As a result, frequency selection significantly impacts data creation and training.

The Number of Crack Defects is proportional to the number of planned layers. Since the wave response coefficient was first computed using the semi-finite element approach, the crack's thickness is proportional to the thickness of each pipeline layer divided by the finite element method.

The Input Circumferential Wave Number (ICWN) value affects the wave propagation curve selection, and different curves are acceptable for different frequency sizes.

The Crack Length in the Radial Direction (CDRD) is proportional to the layer count, and the depth of each layer is equal to the reciprocal of the overall layer count. This characteristic primarily indicates the depth of the cylinder's crack.

Crack Width in the Circumferential Direction (CWCD) has a value range of [0.025,1], classified into 41 groups. The value of CWCD denotes the radian, which primarily indicates the circumferential length of the cylindrical fracture.

Table 1 Setting Information about Fortran Code

Setting Options	Setting Values	Unit
Total Number of Element	40	1
Number of the Materials	[1,6]	1
Thickness Over Mean Radius	[0.1,0.4]	0.1
Circular Frequency	[1,4]	1
Crack Number	41	1
Circumferential Wave Number	[-11,11]	1
Crack Depth in Radial Direction	[0.025,1]	0.025
Crack Width in Circumferential Direction	[0.025,1]	0.025

3.1.2 Settings of the Machine Learning Hyperparameters

The three most often used approaches for hyperparameter optimization are manual, machine-assisted, and algorithm-based. Practicality is the primary factor guiding hyperparameter optimization. The hyperparameter with the best performance is chosen by comparing it to the model's anticipated performance. The hyperparameter settings were discovered after multiple tests and are listed in Table 2.

Table 2 Hyperparameters of Machine Learning Algorithms

Algorithms	**Hyperparameters**	**Value**
Support Vector Machine	Penalty Coefficient (C)	1
	gamma	0.1
K-Nearest Neighbors	K	5
	Feature Weight (w)	Distance
	Distance Metric (p)	1
Random Forest and Extremely Randomized Tree Algorithm	The number of Largest Weak Learners (n_estimators)	100
	Out of Bag (oob_score)	false
	Maximum Number of Features (max_features)	200
	Minimum Number of Samples Required for Subdividing Internal Nodes (min_samples_split)	10
	Minimum Number of Samples for Leaf Nodes (min_samples_leaf)	3

3.1.3 Settings of Deep Learning Hyperparameters

The learning rate is a critical hyperparameter in deep learning. A high or low learning rate may result in an extremely slow or even non-existent model learning speed. The learning rate must be chosen while keeping an eye on the loss function's score. The most likely outcome is that the score fluctuates, but the overall trend is downward. Although the score fluctuated, the overall trend of deterioration was the most logical conclusion. The hyper-parameter parameters provided in Table 3 are determined after multiple tests.

Table 3 Hyperparameters of Deep Learning Algorithms

Algorithms	**Hyperparameters**	**Value**
Recurrent Neural Network	Number of Eochs (num_epochs)	200
	Batch size (batch_size)	6
	Learning Rate (INIT_LR)	0.001
	Number of Layer (num_layer)	3
	Hidden Size (hidden_size)	20
Long-Short Term Memory	Number of Eochs (num_epochs)	100
	Batch size (batch_size)	4
	Learning Rate (INIT_LR)	0.001
	Number of Layer (num_layer)	3
	Hidden Size (hidden_size)	40
	Number of Eochs (num_epochs)	200
	Batch size (batch_size)	20
	Learning Rate (INIT_LR)	0.0001

Gated Recurrent Unit	Number of Layer (num_layer)	3
	Hidden Size (hidden_size)	40

3.2 Assessment Criteria

Two assessment metrics, Mean Absolute Error (MAE) and Coefficient of Determination (R^2), were used to compare the performance of machine learning and deep learning. The Root Mean Square Error (RMSE) was chosen as the loss function's y value.

3.2.1 Mean Absolute Error

The Mean Absolute Error (MAE) is a frequently used loss function in regression models. It quantifies the average modulus length of the projected value error without considering direction. It has a data range of 0 to infinity. The following is the calculating formula:

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n}\left|y_i - \hat{y}_i\right| \tag{3.1}$$

Here, y_i represents the true observed value and $\hat{y}_i$ represents the predicted value. In the same prediction target, the smaller MAE value indicates a better prediction of the model. Conversely, a larger MAE value indicates a worse prediction.

3.2.2 Coefficient of Determination

The Coefficient of Determination (R^2) is a statistical indicator used to represent the regression model and explain the change in the dependent variable's dependability. R^2 is a numerical feature used to define the relationship between two random variables. This assessment criteria index is the most accurate representation of the linear regression approach, and it is calculated as follows:

$$R^2 = 1 - \frac{\sum_i \frac{(y_i - \hat{y}_i)^2}{n}}{\sum_i \frac{(y_i - \bar{y})^2}{n}} \tag{3.2}$$

Here, y^2 indicates the average value of the true observations, the denominator refers to the variance, and the numerator refers to the Root Mean Squares Error (RMSE). Generally, the higher the R^2, the better the prediction result. When R^2 is 1, the predicted value and the true value in the sample are completely equal without any errors. If R^2 is 0, each predicted value of the sample is equal to the average value.

3.2.3 Root Mean Square Error

The Root Mean Square Error (RMSE) is frequently used to quantify the average size of an error; its value is equal to the square root of the average squared difference between the predicted and observed values. The following is the formula:

$$\mathrm{RMSE} = \sqrt{\frac{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}{n}} \tag{3.3}$$

In comparison to MAE, RMSE is a more accurate representation of the sample's outliers. On the contrary, MAE is robust and considers outliers to be damaged data. Generally, the MAE has a smaller expected value than the RMSE.

3.3 Prediction of Pipeline Crack Information

Numerous training sets are used, and the various types of data created by change input variables have varying implications on the prediction outcomes. The data type Control Group serves as the fundamental reference. Case A through Case D all adjust the data type of a particular data variable in the case of control variables. Table 4 contains examples of these data types.

Table 4 Training Data Case

	CFP	ICWN	TOMR	PM
Control Group	1	0	0.1	Composite
Case A	[1, 4]	0	0.1	Composite
Case B	1	[-10,0) ∪ (0,10]	0.1	Composite
Case C	1	0	[0.1, 0.4]	Composite
Case D	1	0	0.1	Isotropic

The table's abbreviations relate to the List of Abbreviations, and to eliminate variations in the amount of data that could affect prediction accuracy, all cases have the same number of data sets, 1640 group sets.

CDRD values are fixed between [0, 1] in the Control Group. The CDRDs are evenly distributed into 41 groups, with a variation of 0.025 between each set. Each CDRD corresponds to one of the 41 CWCD groups. Similarly, CWCD is divided into 41 equal groups and falls inside the interval [0, 1]. The purpose of establishing the Control Group is to enable a more direct comparison of the effect of data modifications on the accuracy of the pipeline crack prediction.

After establishing the Control Group, the goal of Case A was to investigate the effect of CFP (Circular Frequency in Pipeline) modifications on prediction accuracy. The CFP-based data type is separated into two files, one for CDRD prediction and one for CWCD prediction. The first file sets the CWCD to a constant value of 0.5. Similarly, the CDRD value is set to 0.5 in the second file.

Case B is based on data sets containing ICWN (Input Circumferential Wave Number) values. Creating data Case B aims to investigate the effect of changing the ICWN on prediction accuracy. Additionally, Case B is broken into two files for training purposes.

In Case C, a change in TOMR (Thickness Over Mean Radius) indicates a change in the pipe's thickness. Since an overly large TOMR value has no engineering significance, the value is limited to [0.1, 0.4].

Case D considers a material change to the pipeline, substituting composite materials with isotropic materials, intending to examine the effect of different materials on prediction accuracy. It is composed of five different materials, including steel, copper, aluminum, magnesium, and titanium.

3.3.1 Prediction Results of Crack Depth in Radial Direction (CDRD)

Crack Depth in Radial Direction (CDRD) is predicted using four machine learning methods and three deep learning algorithms. Table 5 shows the following outcomes when MAE and R^2 are used as assessment criteria:

Table 5 Using different data types and different algorithms to predict CDRD

	Control Group		**CFP**		**HR**		**ICWN**		**PM**	
	MAE	**R^2**	**MAE**	**R^2**	**MAE**	**R^2**	**MAE**	**R^2**	**MAE**	**R^2**
SVM	0.08963	0.91167	0.12323	0.82835	0.08436	0.92784	0.11651	0.85249	0.10009	0.88513
KNN	0.06043	0.95926	0.06533	0.95127	0.03542	0.98689	0.06601	0.94776	0.06215	0.95394
RF	0.07341	0.93986	0.06210	0.95444	0.02656	0.99171	0.03507	0.98584	0.05301	0.96886
ERT	0.06710	**0.94938**	0.04199	**0.98072**	0.01492	**0.99741**	0.03361	**0.98605**	0.03992	**0.98159**
SRNN	0.09248	0.90741	0.11976	0.89705	0.08856	0.95226	0.04125	0.97623	0.08452	0.92206
LSTM	0.10784	0.88266	0.07862	0.94328	0.03464	0.98109	0.04124	0.97629	0.06754	0.94254
GRU	0.07884	0.92901	0.05962	0.96312	0.04462	0.97483	0.03754	0.98282	0.06141	0.95585

The table clearly indicates that the Extremely Randomized Tree (ERT) method has the highest prediction accuracy in machine learning, while the Gated Recurrent Unit algorithm has the highest prediction accuracy in deep learning (GRU). The algorithm has the highest assessment score across all data categories for prediction results.

The CDRD Prediction Histogram depicted in Figure 13 visually represents the prediction performance. While both MAE and R^2 might reflect prediction performance, a lower MAE number indicates better performance, whereas a higher R^2 value indicates better prediction performance. As a result, Figure 13 simply compares the R^2 approach. After calculations, Table 6 is generated, indicating that the dataset based on Thickness Over Mean Radius (TOMD) has the lowest standard deviation. This suggests that data sets derived from human resources are the most stable when used to predict CDRD.

Table 6 Standard Deviation of Different CDRD Data Types

	Control Group	**CFP**	**HR**	**ICWN**	**PM**
Standard Deviation	0.02473	0.04826	**0.02291**	0.04491	0.0228

As illustrated in Figure 13, after altering the data type, the accuracy of all forecasts is greater than the Control Group's score. This demonstrates that altering the data format has an effect on the accuracy of the CDRD prediction.

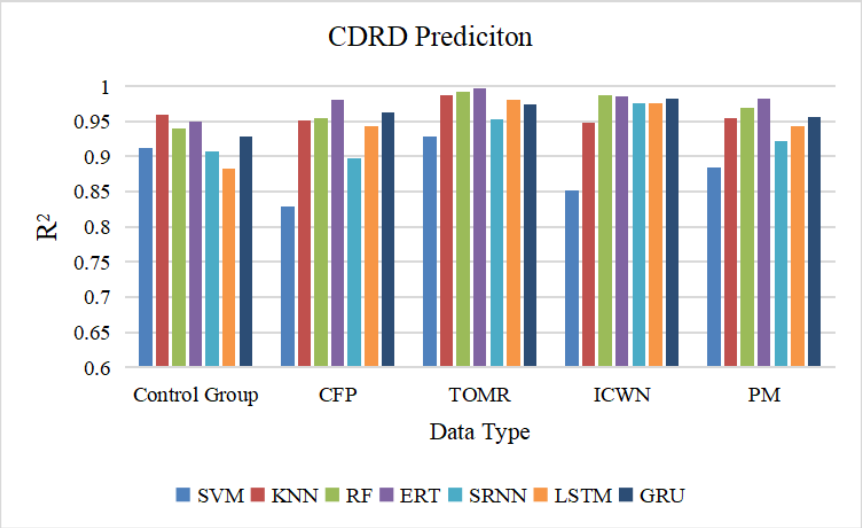

Fig. 13 CDRD Prediction Histogram

3.3.2 Prediction Results of Crack Width in Circumferential Direction (CWCD)

To predict the Crack Width in Circumferential Direction (CWCD), the MAE and R^2 obtained by all algorithms are recorded in Table 7 below.

Table 7 Using different data types and different algorithms to predict CWCD

	Control Group		**CFP**		**HR**		**ICWN**		**PM**	
	MAE	**R^2**	**MAE**	**R^2**	**MAE**	**R^2**	**MAE**	**R^2**	**MAE**	**R^2**
SVM	0.14369	0.74237	0.18478	0.61271	0.18089	0.63239	0.14428	0.7608	0.13541	0.76878
KNN	0.05255	**0.96818**	0.09881	0.87956	0.10823	0.8613	0.05964	0.95549	0.0623	0.95641
RF	0.06952	0.94467	0.05342	0.96713	0.04513	0.97804	0.07264	0.94013	0.05063	0.97251
ERT	0.05736	0.96506	0.02071	**0.99506**	0.02905	**0.99079**	0.05013	**0.97182**	0.02713	**0.9913**
SRNN	0.11124	0.81812	0.09114	0.88614	0.15421	0.75134	0.18421	0.60874	0.09451	0.89012
LSTM	0.10545	0.87654	0.06457	0.93751	0.09974	0.88054	0.11246	0.82275	0.03144	0.98325
GRU	0.07451	0.93961	0.05974	0.9493	0.09424	0.88835	0.08475	0.87642	0.02147	0.99524

As shown in Table 7, the ERT algorithm has the most robust prediction performance in most data situations. The best machine learning algorithm is still ERT, while GRU is the best deep learning approach.

The standard deviation of the scores associated with each data type's prediction outcomes is calculated, as shown in Table 8.

Table 8 Standard Deviation of different CWCD data types

	Control Group	**CFP**	**HR**	**ICWN**	**PM**
Standard Deviation	0.07951	0.11939	0.11691	0.12052	**0.07617**

The distinction from forecasting CDRD is that when Pipeline Materials (PM) data is used to forecast CWCD, it has a better level of stability and predictive performance. The comparison histogram of the CWCD R^2 is shown in Figure 14.

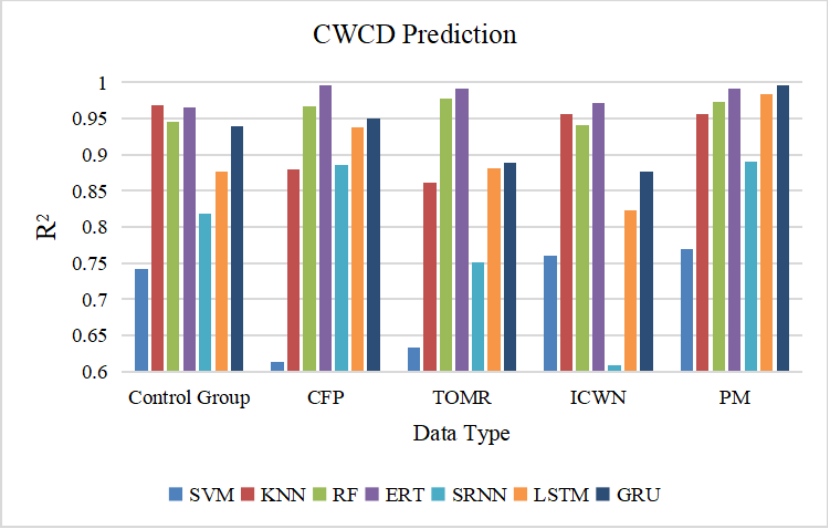

Fig. 14 CWCD Prediction Histogram

The CWCD score graph demonstrates that some algorithms perform worse than the CDRD prediction histogram when predicting CWCD.

Although the Control Group outperformed most of the data types in terms of score stability, Figure 14 shows that when the RF and ERT algorithms are utilized, the diverse data types still have higher scores than the Control Group's prediction results. The majority of methods outperform the Control Group, particularly when PM-based data is used, demonstrating that changing the data type enhanced the CWCD's prediction ability.

3.3.3 Results Comparison of CDRD and CWCD

The R^2 of the CDRD and the CWCD are compared in Figure 15. It is discovered that, in most circumstances, the predictive performance of the CDRD and CWCD are comparable, but the CDRD forecast is more consistent than the CWCD forecast.

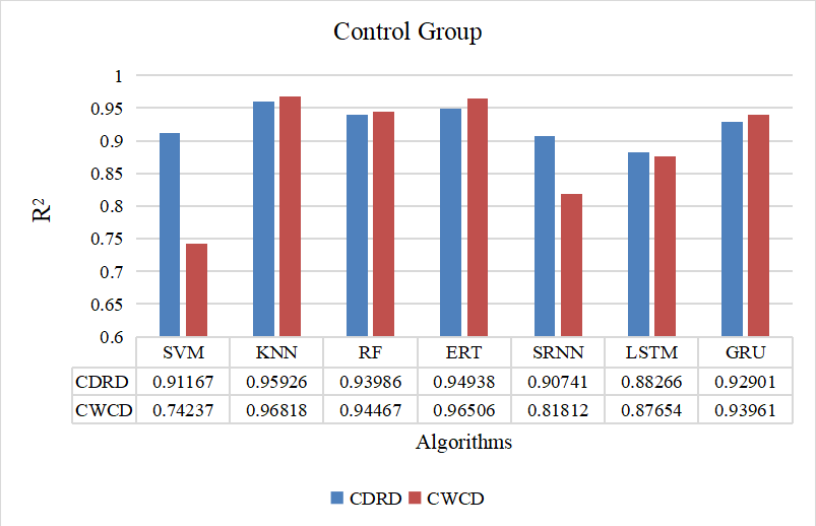

Fig. 15 Performance comparison histogram of CDRD and CWCD in Control Group

In most cases, the CDRD outperforms the CWCD in Figure 16. However, in some algorithms, the result score for CWCD prediction is greater than for CDRD prediction. This could be because the algorithm randomly selects a subset of the data as the test set. For those results

which CWCD has a higher score, the score of CDRD prediction may exceed CWCD prediction after some repeated calculations.

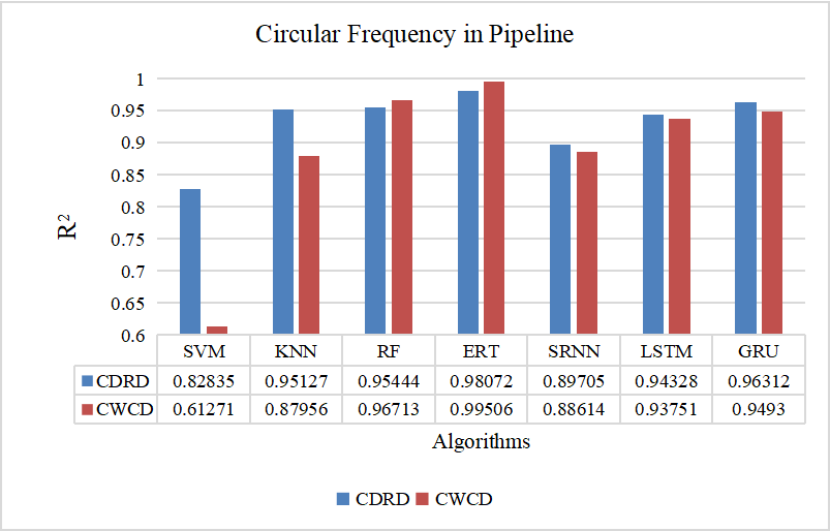

Fig. 16 Performance comparison histogram of CDRD and CWCD in CFP Group

Figure 17 compares the R^2 under the condition of TOMR based data set. Under this data type, the prediction performance of various algorithms for CDRD is better than that for the CWCD.

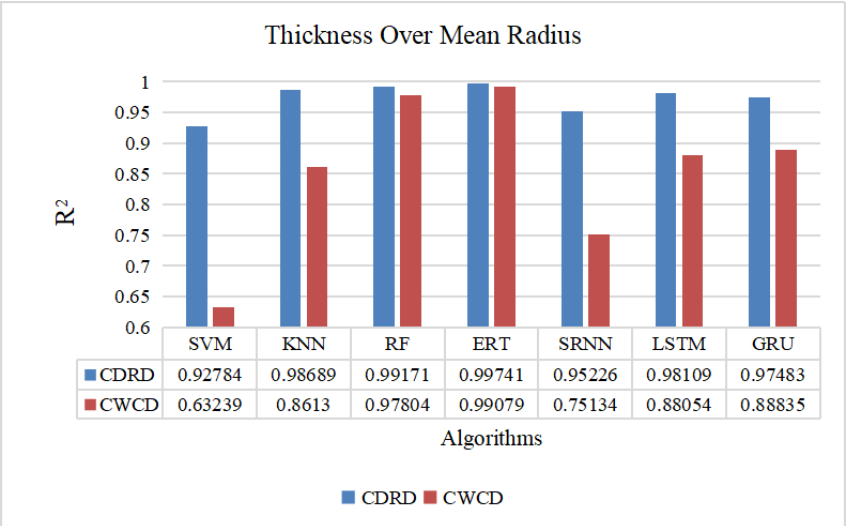

Fig. 17 Performance comparison histogram of CDRD and CWCD in TOMR Group

In Fig.18, the R^2 score for all algorithms that predict CDRD is greater than the R^2 score for algorithms that predict CWCD. Notably, the scores of deep learning algorithms are poor when predicting CWCD, indicating that deep learning methods are not suitable for predicting CWCD utilizing ICWN-based data types.

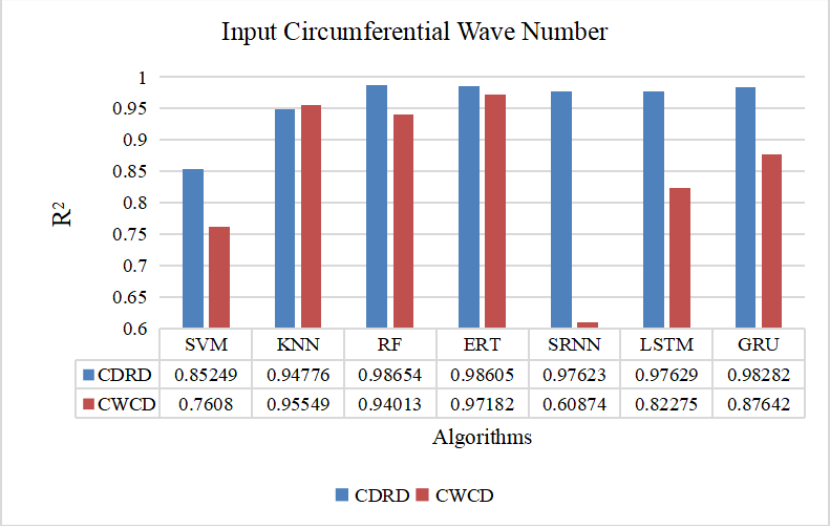

Fig. 18 Performance comparison histogram of CDRD and CWCD in ICWN Group

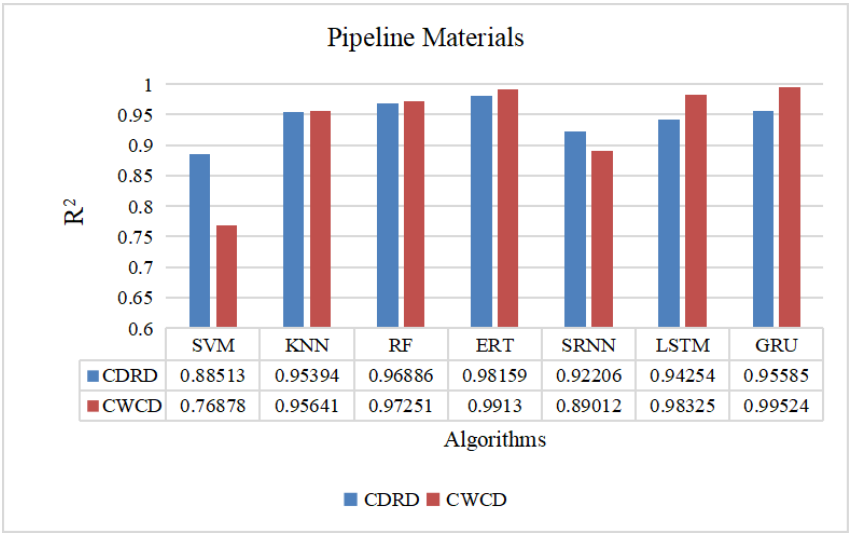

Fig. 19 Performance comparison histogram of CDRD and CWCD in PM Group

As illustrated in Figure 19, the CWCD prediction result score for most algorithms is more than the CDRD prediction result score, and the majority of approaches have an R^2 greater than 0.9. One could argue that the PM-based data type is more suited to forecasting the CWCD.

3.3.4 Comparison of Machine Learning and Deep Learning

"TOTAL" indicates that pooled all the data sets, thereby increasing the sample size. This study collected all data types and utilized seven algorithms to predict CDRD and CWCD. Figure 20 depicts R^2 score histogram comparisons. All the data types in Figure 20 used to share the same data volume, 8200. Compared to the Control Group data type, the decrease of CWCD prediction can be seen clearly, Since the "TOTAL" data sets which provided had different feature distributions, which increases the challenge of training, and increasing the quantity of data did not affect the prediction performance of any method for CWCD. Additionally, all algorithms' prediction performance for the CDRD is superior to CWCD.

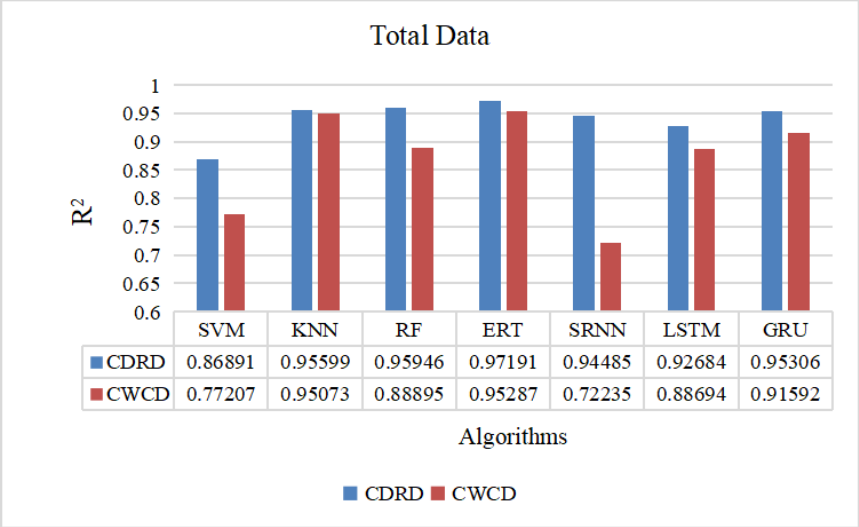

Fig. 20 Performance comparison histogram of CDRD and CWCD in Total Group

To compare the prediction performance of deep learning and machine learning algorithms for the depth and width of pipeline cracks, this study chose the GRU with the highest prediction performance in the deep learning algorithm and the ERT with the highest prediction performance in the machine learning algorithm.

Figures 21 and 22 illustrate different comparisons of the prediction performance of the GRU and ERT algorithms. As seen in the figures, the GRU performs similarly to the ERT algorithm when predicting the CDRD but outperforms the ERT method only when predicting the CWCD.

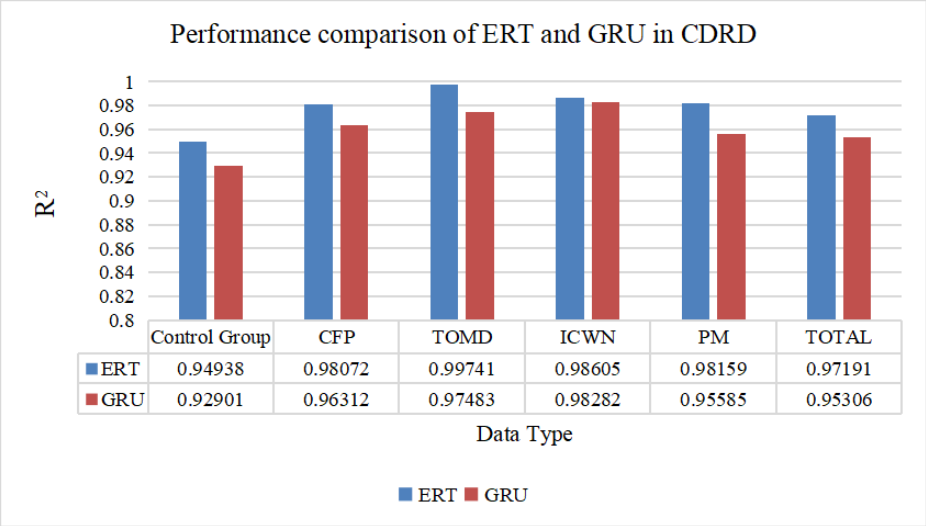

Fig. 21 R^2 comparison histogram

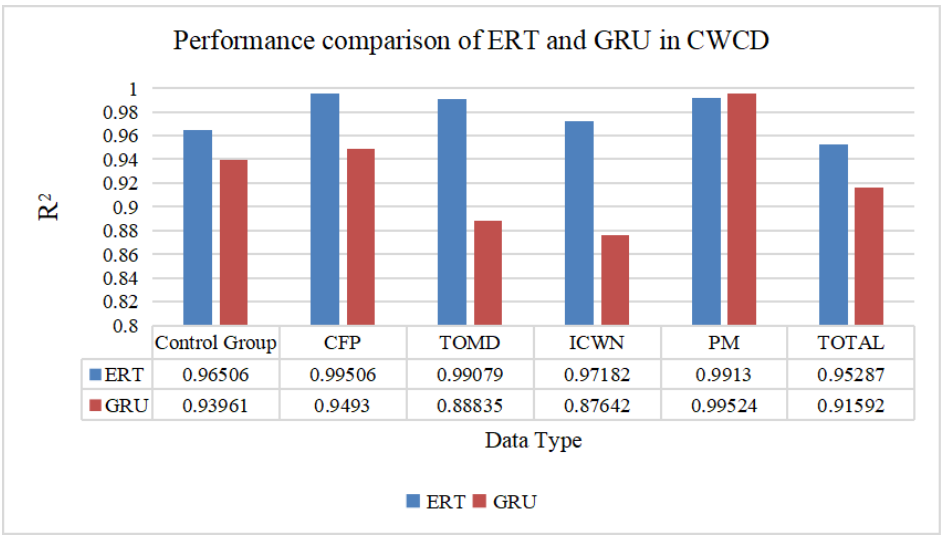

Fig. 22 R^2 comparison histogram

Due to the fact that several hyperparameters were changed during prediction to ensure that no overfitting or underfitting occurred, a loss curve was constructed, as illustrated in Figures 23 and 24. The RMSE was used as the assessment criterion to compare data with aberrant values.

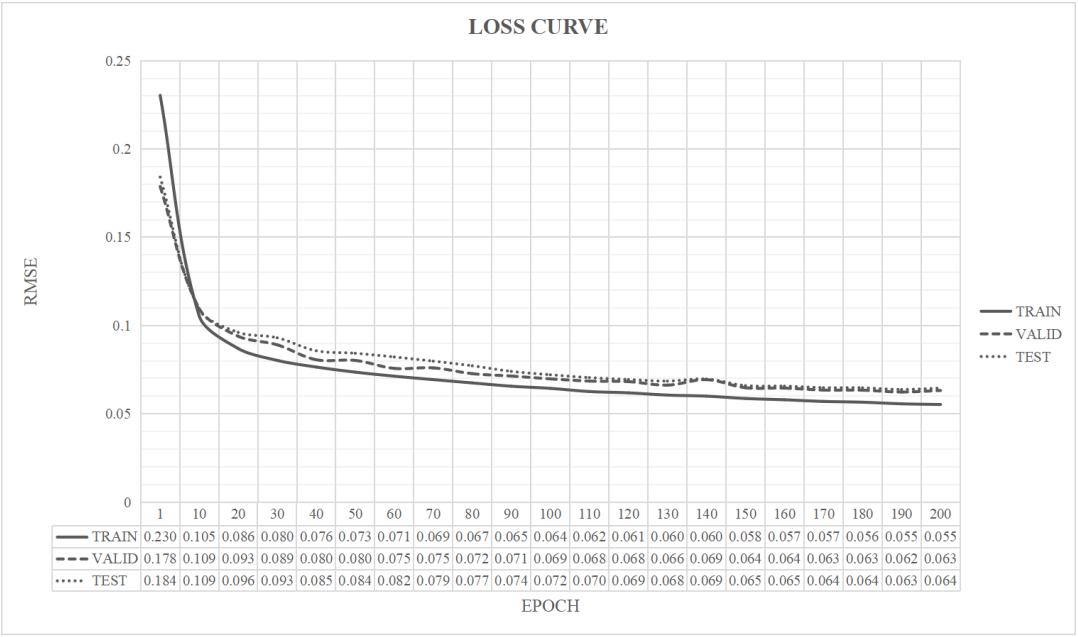

Fig. 23 GRU loss curve in CDRD prediction

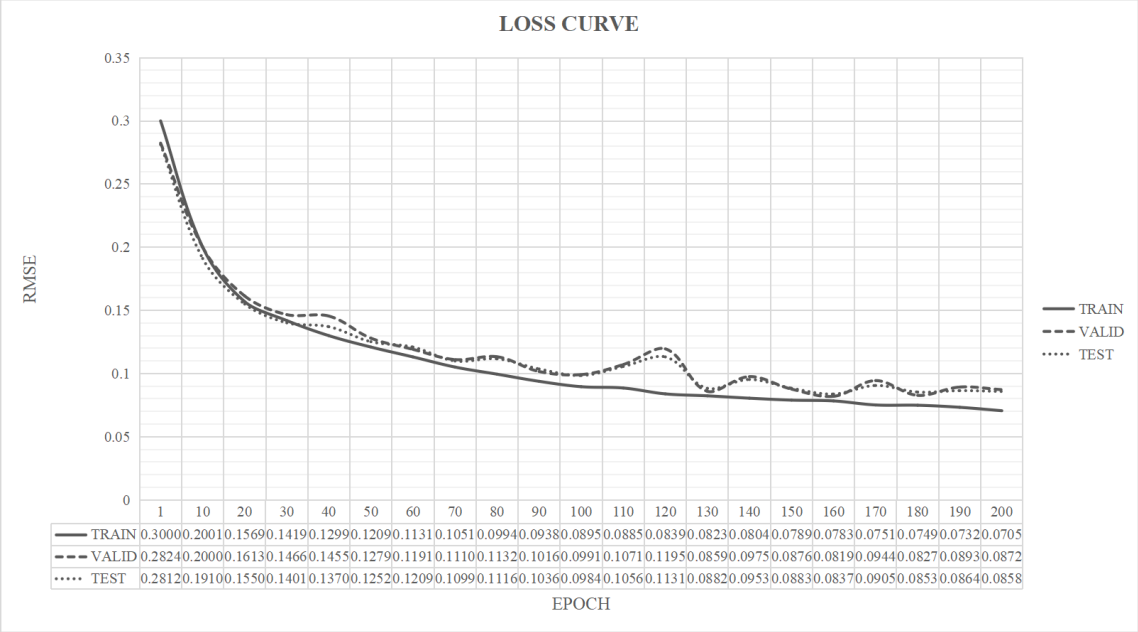

Fig. 24 GRU loss curve in CWCD prediction

As illustrated in Figures 23 and 24, the drop curve exhibits the features of a good fit, indicating no over-or under-fitting in this experiment.

By combining the comparison scores in Figures 21 and 22, it is possible to deduce that when the amount of data is sufficient, the GRU method can approach the prediction performance of the ERT algorithm. Because the predictive performance of a deep learning algorithm depends on the number of datasets and the settings of hyperparameters, one may argue that the GRU can outperform ERT methods in terms of predictive performance.

3.4 Results & Discussion

Throughout Section 3.3, it was discovered that altering the type of datasets affected the algorithms' prediction performance. For example, TOMR data is better suited to forecasting the CDRD, while PM data is better suited to predicting the CWCD. Additionally, increasing the number of datasets improves deep learning algorithms' prediction performance to a certain level. However, the pipeline's TOMR and PM values are constant in more realistic situations, making it extremely difficult to gather sufficient datasets by varying the TOMR and PM. Additionally, it is

not straightforward to expand the number of datasets used to train deep learning systems to improve their prediction performance. As a result, the outcome score achieved by modifying the CFP and ICWN data types is critical.

By combining Figures 15, 16, and 18, it is found that when the data type is changed, the algorithms' prediction performance improves, and the algorithms' prediction performance about CDRD is superior to that of CWCD.

From a technical standpoint, when the data types are varied and the number of datasets is not fixed, the ERT retains an excellent predictive performance. As a result, the machine learning ERT technique is unquestionably the most reliable.

However, from the perspective of developing a pipeline trainer, collecting sufficient datasets based on TOMR or PM would aid in predicting the CDRD and CWCD. However, it is difficult to collect large field data sets, and it is also vital to ensure that these data sets contain just changes in pipeline thickness or material.

Chapter IV Conclusion

This chapter employs seven neural network methods suitable for regression issues, four of which are Machine Learning methods and three of which are Deep Learning algorithms. The combination of machine learning and deep learning with 4 types of data which generated by Fortran software could predict two types of pipeline cracks, including Crack Depth in the Radial Direction (CDRD) and Crack Width in the Circumferential Direction (CWCD). Through histogram comparison, it was shown that the machine learning algorithm Extreme Randomized Tree (ERT) has the best prediction performance. Changing the type of training dataset enhances some algorithms' prediction performance. Choosing data types based on the Circular Frequency in Pipeline (CFP) and the Input Circumferential Wave Number (ICWN) can help enhance prediction performance under realistic settings. In theory, if sufficient data types pertaining to Pipeline Materials (PM) can be gathered, it would be advantageous to forecast the outcome of the CWCD. Gated Recurrent Units (GRU) are the optimal algorithm for deep learning. Increasing the amount of training data and adjusting the hyperparameter settings can assist enhance the prediction performance of a deep learning algorithm. However, the outcomes score

corroborates Tripathi et al. [6].'s assertion that deep learning is not always more favorable than machine learning.

Nonetheless, after adjusting the hyperparameters and increasing the amount of data, the prediction accuracy of the deep learning algorithm for pipeline cracks dramatically improved compared to other data sets with fewer data points. Theoretically, it demonstrates that when sufficient data and a better-matched hyperparameter configuration are available, the prediction accuracy of the deep learning algorithm can exceed that of the machine learning Extremely Randomized Tree (ERT). Collecting massive amounts of data is time-consuming, but it is necessary for developing a more reliable and accurate pipeline crack prediction trainer.

Chapter V Future Works

The Crack Depth in Radial Direction (CDRD) and the Crack Width in Circumferential Direction (CWCD) has been predicted in this study. Thus, it is possible that the Crack Thickness in Axial Direction (CTAD) of the pipeline cracks may still be anticipated, allowing for the establishment of three-dimensional space through the technology. The prediction of 3D crack may be proposed in the future, which could more intuitively describe the characteristics of the cracks.

On the other hand, field data is important, so in the follow-up work, it is worth to collect field data and test them. Because in practical cases, response signals may be polluted or overwhelmed by signal noise, it will decrease the accuracy of prediction. So, filtering techniques may need to be used in subsequent work.

What's more, to obtain a better accuracy, deep Learning usually requires huge data sets to train. The data sets collected, however, are not enough for us to reveal the high-performance of deep learning, so, collecting more data sets might be considered in the future works.

Reference

[1] C. Tschope, E. Schulze, H. Neunubel, M. Wolff, R. Schubert and R. Hoffmann, "Experiments in acoustic structural health monitoring of airplane parts," 2008 IEEE International Conference on Acoustics, Speech and Signal Processing, 2008, pp. 2037-2040, doi: 10.1109/ICASSP.2008.4518040.

[2] Jiang Chengjun, Ju Ximin. The Development and Actuality of Oil& Gas Pipeline Testing [J]. Inner Mongolia Petrochemical Industry, 2008(3): 83-86.

[3] A.A. Carvalho, J.M.A. Rebello, M.P.V. Souza, L.V.S. Sagrilo, S.D. Soares, Reliability of non-destructive test techniques in the inspection of pipelines used in the oil industry, International Journal of Pressure Vessels and Piping, Volume 85, Issue 11, 2008, Pages 745-751, ISSN 0308-0161, https://doi.org/10.1016/j.ijpvp.2008.05.001.

[4] Zhao Caiping. Pipeline ultrasonic guided wave inspection data analysis and defect diagnosis system development. (Doctoral dissertation, Beijing University of Technology).

[5] Herrera, Roberto & Christensen,

Paul & Elvers, Adrianus. (2019). Machine Learning in Pipeline Inspection: Applications of supervised learning in non-destructive evaluation.

[6] Tripathi, G.; Anowarul, H.; Agarwal, K.; Prasad, D.K. Classification of Micro-Damage in Piezoelectric Ceramics Using Machine Learning of Ultrasound Signals. Sensors 2019, 19, 4216. https://doi.org/10.3390/s19194216

[7] A. Mardanshahi, V. Nasir, S. Kazemirad, M.M. Shokrieh, Detection and classification of matrix cracking in laminated composites using guided wave propagation and artificial neural networks, Composite Structures, Volume 246, 2020, 112403, ISSN 0263-8223,

[8] F. Zhang, K. Pinkal, P. Wefing, F. Conradi, J. Schneider and O. Niggemann, "Quality Control of Continuous Wort Production through Production Data Analysis in Latent Space," 2019 IEEE International Conference on Industrial Technology (ICIT), 2019, pp. 1323-1328, doi: 10.1109/ICIT.2019.8755111.

[9] Te Ma, Satoru Tsuchikawa, Tetsuya Inagaki, Rapid and non-destructive seed viability prediction using near-infrared hyperspectral imaging coupled with a deep learning approach, computers and Electronics in Agriculture, Volume 177, 2020, 105683, ISSN 0168-1699, https://doi.org/10.1016/j.compag.2020.105683.

[10] Burlina P, Billings S, Joshi N, Albayda J (2017) Automated diagnosis of myositis from muscle ultrasound: Exploring the use of machine learning and deep learning methods. PLoS ONE 12(8): e0184059. https://doi.org/10.1371/journal.pone.0184059

[11] Xiong Zhengsi, Huang Gang, Hao Lijun, etc. Research on non-invasive detection of liver cancer based on machine learning [J]. Beijing Biomedical Engineering, 2020, 39(1): 74-79.

[12] William Sorteberg, Stef Garasto, Alison Pouplin, Chris Cantwell, and Anil A. Bharath. Approximating the Solution to Wave Propagation using Deep Neural Networks. In NeurIPS Workshop on Modeling the Physical World: Perception, Learning, and Control, December 2018.

[13] Rautela M. and Gopalakrishnan S. Deep Learning frameworks for wave propagation-based damage detection in 1D-waveguides, 11th International Symposium on NDT in Aerospace, Paris. Nov.2019.

[14] Yohei Nishizaki, Matias Valdivia, Ryoichi Horisaki, Katsuhisa Kitaguchi, Mamoru Saito, Jun Tanida, and Esteban Vera, "Deep learning wavefront sensing," Opt. Express 27, 240-251 (2019)

[15] P. Zhu, J. Isaacs, B. Fu and S. Ferrari, "Deep learning feature extraction for target recognition and classification in underwater sonar images," 2017 IEEE 56th Annual Conference on Decision and Control (CDC), 2017, pp. 2724-2731, doi: 10.1109/CDC.2017.8264055.

[16] Rymarczyk T, Kłosowski G, Kozłowski E. A Non-Destructive System Based on Electrical Tomography and Machine Learning to Analyze the Moisture of Buildings. Sensors (Basel). 2018;18(7):2285. Published 2018 Jul 14. doi:10.3390/s18072285

[17] Datta, S.K., & Shah, A.H. (2009). Elastic Waves in Composite Media and Structures: With Applications to Ultrasonic Nondestructive Evaluation (1st ed.). CRC Press. https://doi.org/10.1201/9780429136696

[18] Nesreen K. Ahmed, Amir F. Atiya, Neamat El Gayar & Hisham El-Shishiny (2010) An Empirical Comparison of Machine Learning Models for Time Series Forecasting, Econometric Reviews, 29:5-6, 594-621, DOI: 10.1080/07474938.2010.481556

[19] I. Jahan and S. Z. Sajal, Stock Price Prediction using Recurrent Neural Network Algorithm on Time-Series Data, the Midwest Instruction and Computing Symposium 2018, April 6-7, 2018 Duluth MN, USA.

[20] F. Altché and A. de La Fortelle, "An LSTM network for highway trajectory prediction," 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC), 2017, pp. 353-359, doi: 10.1109/ITSC.2017.8317913.

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

[22] J.L.Elman, Finding structure in time, Cogn. Sci. 14(1990)179-211.

Appendices

Appendix A: Control group file about CDRD

Table 9 Control group file about CDRD

CDRD	CWCD	trscoe1	trscoe2	trscoe3	trscoe4	…	trscoe46	refcoe1	refcoe2	refcoe3	refcoe4	…	refcoe46
0	0.05	1	0	0	0	…	0	0	0	0	0	…	0
0.025	0.05	1.000273	0.000153	0.000272	0.00003	…	0.000251	0.000274	0.000155	0.000271	0.00003	…	0.000249
0.05	0.05	1.006377	0.005495	0.010225	0.001167	…	0.009747	0.00954	0.005116	0.009429	0.001094	…	0.009062
0.075	0.05	1.000742	0.000482	0.000966	0.000093	…	0.000854	0.00078	0.000455	0.000771	0.000081	…	0.000684
0.1	0.05	1.001104	0.000719	0.001529	0.000139	…	0.001336	0.001216	0.000675	0.001203	0.000119	…	0.001048
0.125	0.05	1.001149	0.000806	0.001872	0.00016	…	0.001622	0.001286	0.000689	0.001272	0.000119	…	0.001086
0.15	0.05	1.001236	0.000985	0.002336	0.000195	…	0.002022	0.001342	0.000755	0.001328	0.000124	…	0.001116
0.175	0.05	1.001567	0.001415	0.003252	0.000271	…	0.002817	0.00169	0.001065	0.001673	0.000163	…	0.001407
0.2	0.05	1.003369	0.002398	0.004569	0.000428	…	0.00408	0.001811	0.000226	0.0018	0.000046	…	0.001314
0.225	0.05	1.002482	0.002206	0.004917	0.000408	…	0.004285	0.002077	0.001192	0.002063	0.000165	…	0.001616
0.25	0.05	1.002465	0.002438	0.005407	0.000446	…	0.004722	0.001981	0.001292	0.00197	0.000172	…	0.001513
0.275	0.05	1.002624	0.002752	0.006083	0.000496	…	0.005318	0.002128	0.001447	0.002118	0.000185	…	0.001594
0.3	0.05	1.002871	0.00312	0.006905	0.000553	…	0.006037	0.002432	0.00164	0.002421	0.000202	…	0.001795
0.325	0.05	1.002996	0.003411	0.00755	0.000599	…	0.006612	0.002572	0.001796	0.00256	0.000218	…	0.001869
0.35	0.05	1.002989	0.003639	0.008026	0.000635	…	0.007051	0.002524	0.001977	0.002511	0.000242	…	0.001808
0.375	0.05	1.002663	0.004117	0.008725	0.000701	…	0.007703	0.002576	0.002832	0.002559	0.000342	…	0.001981
0.4	0.05	1.003597	0.003931	0.008879	0.00069	…	0.007827	0.002544	0.001434	0.002533	0.000197	…	0.001654
0.425	0.05	1.003535	0.004182	0.009505	0.000734	…	0.008391	0.002609	0.001866	0.002601	0.000252	…	0.001706
0.45	0.05	1.003537	0.0043	0.00993	0.00076	…	0.008777	0.002646	0.00203	0.002642	0.000281	…	0.001729
0.475	0.05	1.003581	0.00425	0.010127	0.000777	…	0.008966	0.002458	0.002034	0.002462	0.0003	…	0.001558

0.5	0.05	1.004379	0.005747	0.011275	0.000932	…	0.010269	0.001215	0.001967	0.001233	0.000377	…	0.000176
0.525	0.05	1.002148	0.005103	0.010174	0.000833	…	0.008993	0.002795	0.004555	0.002786	0.000492	…	0.002406
0.55	0.05	1.002077	0.005229	0.011923	0.000974	…	0.010628	0.003123	0.004374	0.003108	0.00052	…	0.002562
0.575	0.05	1.001651	0.005928	0.013533	0.001127	…	0.012089	0.00377	0.005597	0.003751	0.000603	…	0.003197
0.6	0.05	1.003924	0.008697	0.015491	0.001289	…	0.014297	0.002971	0.002185	0.002989	0.00035	…	0.001605
0.625	0.05	1.001686	0.003603	0.011701	0.000534	…	0.009617	0.008683	0.009844	0.008604	0.000671	…	0.007919
0.65	0.05	1.001536	0.007309	0.018405	0.001554	…	0.016768	0.003927	0.004383	0.003928	0.000622	…	0.002904
0.675	0.05	1.002722	0.009873	0.020605	0.001811	…	0.019096	0.00281	0.002405	0.002836	0.000646	…	0.001458
0.7	0.05	1.004137	0.017238	0.024002	0.00231	…	0.022931	0.001882	0.006533	0.001877	0.001004	…	0.001944
0.725	0.05	1.007573	0.021213	0.024564	0.002335	…	0.023889	0.005957	0.010871	0.005887	0.00108	…	0.0061
0.75	0.05	1.008552	0.025715	0.014066	0.001041	…	0.013717	0.016837	0.029928	0.016641	0.001835	…	0.016784
0.775	0.05	0.999071	0.008017	0.024199	0.002214	…	0.022411	0.003267	0.007543	0.003265	0.001206	…	0.002566
0.8	0.05	1.000905	0.012962	0.02716	0.002625	…	0.02577	0.000262	0.002757	0.000286	0.001376	…	0.001292
0.825	0.05	1.003825	0.023203	0.03204	0.003364	…	0.031205	0.005789	0.007398	0.005635	0.001968	…	0.007113
0.85	0.05	1.009373	0.026781	0.032686	0.003426	…	0.032414	0.010014	0.013802	0.009804	0.00196	…	0.011212
0.875	0.05	1.004986	0.02444	0.012433	0.000603	…	0.011827	0.015836	0.036766	0.015711	0.002028	…	0.015791
0.9	0.05	0.997405	0.011603	0.03207	0.003445	…	0.030645	0.007483	0.007577	0.00727	0.002735	…	0.007828
0.925	0.05	0.999319	0.014209	0.035421	0.003997	…	0.034306	0.011179	0.004375	0.010891	0.003196	…	0.011681
0.95	0.05	1.014179	0.031886	0.038319	0.004417	…	0.038834	0.016213	0.021792	0.016002	0.002677	…	0.01834
0.975	0.05	1.004876	0.015964	0.029404	0.002465	…	0.02906	0.00732	0.021236	0.007185	0.00151	…	0.00879
1	0.05	0.99927	0.006975	0.038837	0.004801	…	0	0	0	0	0	…	0
…	…	…	…	…	…	…	…	…	…	…	…	…	…

Appendix B: Control group file about CWCD

Table 10 Control group file about CWCD

CWCD	CDRD	trscoe1	trscoe2	trscoe3	trscoe4	…	trscoe46	refcoe1	refcoe2	refcoe3	refcoe4	…	refcoe46
0	0.05	1	0	0	0	…	0	0	0	0	0	…	0
0.025	0.05	1.004379	0.005747	0.011275	0.000932	…	0.010269	0.001215	0.001967	0.001233	0.000377	…	0.000176
0.05	0.05	1.001169	0.003077	0.016093	0.000736	…	0.009721	0.011757	0.002905	0.011673	0.000114	…	0.00625
0.075	0.05	0.999528	0.003942	0.022684	0.000697	…	0.008741	0.018545	0.004928	0.018288	0.000585	…	0.00459
0.1	0.05	0.993939	0.005214	0.029408	0.00092	…	0.007889	0.030004	0.008709	0.029247	0.001232	…	0.015079
0.125	0.05	0.994274	0.007145	0.040975	0.001278	…	0.010192	0.033827	0.006532	0.032866	0.001013	…	0.004718
0.15	0.05	0.991237	0.007157	0.047383	0.00152	…	0.011431	0.040072	0.007909	0.038609	0.000682	…	0.006023
0.175	0.05	0.990471	0.007806	0.052923	0.001105	…	0.010454	0.046543	0.01008	0.044184	0.001537	…	0.00501
0.2	0.05	0.989817	0.011325	0.064116	0.000307	…	0.010658	0.053125	0.010546	0.049636	0.002729	…	0.002275
0.225	0.05	0.985638	0.01136	0.06996	0.002118	…	0.012126	0.062135	0.01225	0.057029	0.001769	…	0.005941
0.25	0.05	0.983165	0.011584	0.074986	0.002228	…	0.013526	0.068303	0.013405	0.061684	0.001261	…	0.005738
0.275	0.05	0.979251	0.015881	0.084481	0.003783	…	0.009975	0.068797	0.012554	0.061584	0.007209	…	0.023109
0.3	0.05	0.981644	0.015442	0.087945	0.00096	…	0.011154	0.082242	0.016263	0.070612	0.003841	…	0.003696
0.325	0.05	0.978132	0.015771	0.092306	0.003063	…	0.014368	0.090277	0.017724	0.075508	0.001981	…	0.006962
0.35	0.05	0.97589	0.0161	0.095606	0.002559	…	0.016331	0.096985	0.019076	0.078866	0.002067	…	0.006447
0.375	0.05	0.977282	0.019142	0.100817	0.003827	…	0.011416	0.100617	0.018801	0.080327	0.0059	…	0.008018
0.4	0.05	0.974737	0.019887	0.104007	0.000794	…	0.01352	0.11053	0.021652	0.083914	0.004874	…	0.00522
0.425	0.05	0.971958	0.020374	0.105944	0.003205	…	0.017103	0.118605	0.023031	0.086596	0.002385	…	0.007456
0.45	0.05	0.962419	0.025171	0.108637	0.00201	…	0.02234	0.119983	0.02212	0.086857	0.005986	…	0.038071
0.475	0.05	0.972587	0.024007	0.10925	0.008472	…	0.015318	0.129428	0.024142	0.088664	0.008519	…	0.00449
0.5	0.05	0.969347	0.024548	0.11038	0.005038	…	0.017564	0.137971	0.02681	0.088597	0.008859	…	0.005498

0.525	0.05	0.966401	0.024891	0.110188	0.002794	…	0.020918	0.146943	0.02857	0.088542	0.006826	…	0.007198
0.55	0.05	0.966836	0.028126	0.108811	0.000613	…	0.01051	0.149875	0.027227	0.088635	0.004456	…	0.012186
0.575	0.05	0.966973	0.028227	0.10726	0.003986	…	0.013008	0.158609	0.029987	0.087543	0.000869	…	0.00389
0.6	0.05	0.965569	0.029169	0.105328	0.003263	…	0.018153	0.166594	0.031934	0.085398	0.002448	…	0.00871
0.625	0.05	0.957648	0.029972	0.104673	0.005943	…	0.032564	0.176499	0.034761	0.08141	0.010427	…	0.027957
0.65	0.05	0.963602	0.032881	0.098807	0.001169	…	0.010585	0.178506	0.0327	0.08098	0.003427	…	0.007792
0.675	0.05	0.963209	0.032829	0.095346	0.002941	…	0.014029	0.186322	0.035105	0.077515	0.001437	…	0.006074
0.7	0.05	0.962728	0.033755	0.091419	0.002379	…	0.020511	0.194738	0.037166	0.072876	0.002069	…	0.0085
0.725	0.05	0.96337	0.040355	0.081756	0.006886	…	0.012393	0.196099	0.032954	0.072516	0.005732	…	0.024377
0.75	0.05	0.961257	0.037233	0.079373	0.001229	…	0.010424	0.206733	0.038201	0.065095	0.002399	…	0.004188
0.775	0.05	0.961126	0.03752	0.074665	0.002307	…	0.015657	0.214097	0.039919	0.059936	0.001122	…	0.00763
0.8	0.05	0.952582	0.039044	0.072143	0.005129	…	0.038905	0.226221	0.043707	0.052131	0.005043	…	0.031682
0.825	0.05	0.961391	0.042386	0.058223	0.001407	…	0.008969	0.226992	0.041149	0.049442	0.001868	…	0.009334
0.85	0.05	0.959739	0.04177	0.052657	0.001027	…	0.010474	0.234647	0.043169	0.0428	0.001409	…	0.004281
0.875	0.05	0.960432	0.042157	0.047833	0.001352	…	0.017018	0.241897	0.044768	0.0365	0.000963	…	0.007502
0.9	0.05	0.96417	0.045747	0.036264	0.001139	…	0.011453	0.247093	0.044824	0.031485	0.000683	…	0.01241
0.925	0.05	0.960393	0.04637	0.027416	0.000564	…	0.006491	0.255581	0.046764	0.023075	0.000766	…	0.002896
0.95	0.05	0.959423	0.046296	0.022042	0.000482	…	0.010714	0.262032	0.04759	0.016808	0.000676	…	0.006539
0.975	0.05	0.963705	0.046935	0.012189	0.000556	…	0.007296	0.262046	0.049494	0.017328	0.000908	…	0.013368
1	0.05	0.95682	0.049832	0	0	…	0	0.278287	0.050428	0	0	…	0
…	…	…	…	…	…	…	…	…	…	…	…	…	…

Appendix C: CDRD prediction codes in Extremely Randomized Tree

```
from sklearn import datasets                                    # Create the environment
from sklearn import metrics
from sklearn.metrics import r2_score
from sklearn.model_selection import train_test_split
from sklearn.ensemble import ExtraTreesRegressor
from sklearn.preprocessing import StandardScaler                # Preprocessing function
from numpy import *


pipeline_test= pd.read_csv(r"C:\Users\admin\data\CDRD\CDRD.csv")
x=pipeline_test.drop('CDRD',axis=1)
y=pipeline_test['CDRD']
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2)      # f(x) = (x - means) / standard deviation
scaler = StandardScaler()
scaler.fit(x_train)
x_train = scaler.transform(x_train)                             # Standardization
x_test = scaler.transform(x_test)


etr = ExtraTreesRegressor(n_estimators=10, max_features=200,    # Build Extremely Randomized Tree model
min_samples_split=10)
etr.fit(x_train,y_train)
y_predict = etr.predict(x_test)
print(np.sqrt(metrics.mean_squared_error(y_test, y_predict)))   # MAE
print(svr.score(x_test, y_test))                                # R2 Score
```

Appendix D: Part of the CDRD prediction codes in Gated Recurrent Unit

```
xlsfile=pd.read_csv('CDRD TOTAL.csv',header=None)                    # Read data
data=np.array(xlsfile)
n=np.random.randint(0,data.shape[0],data.shape[0])                   # Random array used to scramble the data set


train_data = data[n[0:6500],2:]                                      # Divide data features into training set, validation set and test set
valid_data = data[n[6500:7500],2:]
test_data = data[n[7500:],2:]


train_label = data[n[0:6500],0:2]                                    # Obtain the corresponding labels of the training set, validation set and test set, column 0-1 in the table
valid_label = data[n[6500:7500],0:2]
test_label = data[n[7500:],0:2]


tf.set_random_seed(100)


num_epochs = 200                                                     # Training related hyperparameters
batch_size = 2
alpha = 0.0001
hidden_nodes = 40
input_features = 222
sequence_len = 1
output_class = 2                                                     # Regression 8 input with 4 output
X = tf.placeholder("float", [None, sequence_len, input_features])    # Input placeholder
Y = tf.placeholder("float", [None, sequence_len, output_class])
```

```
weights = {
   'out': tf.Variable(tf.random_normal([hidden_nodes, output_class]))
}
biases = {
   'out': tf.Variable(tf.random_normal([output_class]))
}

def GRU(x):
x = tf.reshape(x , [-1, sequence_len, input_features])
gru_cell1 = tf.nn.rnn_cell.GRUCell(num_units=hidden_nodes)
gru_cell2 = tf.nn.rnn_cell.GRUCell(num_units=hidden_nodes)
gru_cell3 = tf.nn.rnn_cell.GRUCell(num_units=hidden_nodes)

gru_cell = tf.nn.rnn_cell.MultiRNNCell([gru_cell1, gru_cell2, gru_cell3])

init_state = gru_cell.zero_state(tf.shape(x)[0], dtype=tf.float32)

outputs, _ = tf.nn.dynamic_rnn(gru_cell, x, dtype=tf.float32, initial_state=init_state)
output_sequence = tf.matmul(tf.reshape(outputs, [-1, hidden_nodes]), weights['out']) + biases['out']

return tf.reshape(output_sequence, [-1, sequence_len, output_class])
```

```
# Define weights, gaussian distribution

# Define the GRU network
# Reshape input tensor into batch x sequence length x # of features

# 3 GRU with hidden number of nodes each layer

# Stack of those layers

# Initialize state

# Get the output of each state
```

This page is intentionally left blank.