Data and Informatics-informed Digital Twins for Smart and Practical Bridge Maintenance

Yan Gao

School of Engineering

Cardiff University

A thesis submitted to Cardiff University for the degree of Doctor of Philosophy 2023

Acknowledgements

I am deeply grateful to all those who have contributed to the completion of this PhD thesis; without whose support and guidance, this journey would not have been possible.

First and foremost, I would like to express my heartfelt gratitude to my supervisors, Prof. Haijiang Li, Prof. Rossi Setchi, and Dr. Ze Ji, whose invaluable mentorship and expert guidance have been the cornerstone of my research.

I appreciate the engineering manager, Mr. Jarrod Richards, from Centregreat Rail, whose valuable knowledge and experiences have greatly contributed to this research, as well as my colleagues, Guanyu Xiong and Chengzhang Chai, with whom the discussion and shared skills have enriched my study and made this journey more enjoyable.

I want to acknowledge Cardiff University and China Scholarship Council's financial support, which allowed me to pursue this research.

Finally, I sincerely thank my parents for their boundless love and unwavering support throughout my life. Their love and encouragement sustained me during the challenging times. This achievement is as much theirs as it is mine.

Yan Gao

Sep 30, 2023

Abstract

Digital twinning has garnered significant interest for its potential to revolutionize bridge operation and maintenance (O&M) in the era of Industry 4.0. Although digital twin (DT) systems have achieved considerable development in academia and are getting increasingly popular in the industry, many gaps exist between academic research and widespread applications in route practices of bridge O&M, such as DT's efficiency, resiliency, and intelligence, which are associated with different scenarios in a realistic bridge maintenance cycle. This research is conducted to solve the issues encountered in DT practical applications for bridge maintenance, and it can be structured according to five parts.

Firstly, an efficient and resilient bidirectional DT framework is proposed for bridge O&M based on a comprehensive understanding of communication complexity by leveraging AI-informed edge computing, information hierarchy, and low-powered wide area network (LPWAN). The theoretical framework is idealised mathematically with state-space representation and modelled using Petri-net. The related study indicates that the time delay of DT consists of computation and communication time costs and reveals the distinct impact of their sequence on DT latency. Moreover, the framework is further developed into a cross-platform prototype based on embedded systems, long-range wide area network (LoRaWAN), HTTP and MQTT protocols, restful web services, and IFC-based human-machine interface (HMI). The prototype is validated through different scenarios for bridge O&M, including drone-enabled bridge inspection, IoT-based bridge health-state monitoring, and decentralised dynamic evacuation.

Secondly, an improved Prototypical Network (ProtoNet) is proposed for image-based bridge damage detection based on few-shot learning. It can work with only a few annotated examples, avoiding the tedious and labour-intensive data acquisition required by supervised learning. Feature embedding is "training free", achieved through cross-domain transfer learning from ImageNet. The approach is explored on a public dataset through ablation studies and reaches over 94% mean accuracy for 2-way 5-shot classification via the pre-trained GoogleNet. Moreover, the proposed fine-tuning methods are demonstrated with better performance than previous research. Finally, the approach is validated using real bridge inspection images, demonstrating its capability of fast implementation for practical damage inspection.

Thirdly, a highly efficient framework is proposed for spatial damage assessment and DT synchronization based on point clouds in near real-time. The 2D damage is detected via DeepLabV3+ on pseudo grayscale images only from the point depth, avoiding the drawbacks of image and point cloud fusion. Then, 3D damage is separated through voxelization and converted into a lightweight binary matrix that can be further compressed losslessly for DT synchronization. The framework is validated via two case studies, demonstrating that the proposed voxel-based method can be easily applied to real-world damage with non-convex geometry instead of convex-hull fitting; FE and BIM models can be updated automatically through the framework.

Fourthly, an automatic and unified deep learning (DL) framework is proposed for intelligent fault diagnosis (IFD) and health-state recognition based on time series by leveraging automated machine learning (AutoML) and data-level fusion. Uniaxial or triaxial signals can be reconstructed into 3-channel pseudo-images to satisfy the CNN input requirements and achieve data-level fusion simultaneously. Then, the model training, hyperparameter optimisation, and evaluation are carried out automatically. The selected model can be deployed on a cloud server or an edge device. Moreover, the framework can be extended by integrating multi-channel 1D-CNN architectures, validated using the data from a railway bridge's vibration-based monitoring (VBM) project. Multiple sensors' data-level and decision-level fusion performances are also compared and analysed.

Fifthly, a knowledge graph (KG) schema based on bridge structure and maintenance reports is proposed. Then, graph data mining is explored on the established KG by leveraging large language models (LLMs) and graph neural networks (GNNs). The models trained from graph contextual similarity can identify node layer information and provide maintenance recommendation for unsolved defects from the existing options. Moreover, an intact workflow integrating the proposed KG schema and data mining approaches is designed for maintenance routine practices.

Finally, a preliminary bridge DT system is developed based on the above outcomes. This research addresses some challenges encountered for bridge DT implementation in practical O&M and has paved the way for a more efficient, resilient, and intelligent bridge DT system. As such, it offers excellent potential to achieve generalised DT applications in smart and practical bridge maintenance.

Table of contents

Acknowledgements i

Abstract ii

Table of contents v

List of Figures ix

List of abbreviations xiii

List of outcomes xv

Chapter 1 Introduction 1

1.1 Research background 1

1.2 Research motivations 3

1.3 Research questions 4

1.4 Research innovations 5

1.5 Structure of the thesis 6

Chapter 2 Literature review 10

2.1 Bridge DT frameworks and systems 10

2.2 Bridge-machine-human triplet interaction 14

2.2.1 Bridge-machine Interaction 15

2.2.2 Machine-human Interaction 16

2.2.3 Human-bridge Interaction 19

2.3 Vision-based bridge inspection 20

2.3.1 2D image-based survey 20

2.3.2 3D point-cloud based survey 26

2.4 IFD and health monitoring with time series 29

2.4.1 Traditional machine learning pipeline 29

2.4.2 Deep learning pipeline 31

2.4.3 Multi-sensor data fusion 34

2.5 Bridge maintenance knowledge graph 34

2.6 Summary of literature findings 37

Chapter 3 Research Methodology 40

3.1 Research philosophies 40

3.2 Research approaches, strategies and choices 41

3.3 Research time horizons, techniques and procedures 42
3.4 Overarching research roadmap 42
Chapter 4 AIoT-informed DT framework 44
4.1 Theoretical foundation 44
4.2 Time delay and complexity 45
4.2.1 Fault tolerance and topology 51
4.2.2 Mathematical idealization 52
4.2.3 Petri-net modelling 54
4.3 Framework design and development 57
4.3.1 AI-based edge computing 57
4.3.2 LPWAN communication 59
4.3.3 Cloud and protocols 61
4.3.4 Web-based platform 62
4.4 Proof of concept 63
4.4.1 Drone-enabled bridge inspection 64
4.4.2 Vibration-based monitoring 69
4.4.3 Dynamic evacuation 72
4.4.4 Comparative analysis 76
4.5 Summary 77
Chapter 5 Few-shot bridge image damage detection 80
5.1 Theoretical foundation 80
5.1.1 Few-shot problem definition 80
5.1.2 Meta-learning and feature embedding 82
5.1.3 Transfer learning and fine-tuning 83
5.2 Few-shot damage detection approach 84
5.2.1 Proposed architecture 84
5.2.2 Domain adaption and transduction 86
5.2.3 Loss function and fine-tuning 89
5.3 Experiments and approach validation 90
5.3.1 Experiment preparation 90
5.3.2 Ablation studies 91
5.3.3 Few-shot damage detection 99

6.1.1 Summary 102

Chapter 6 Point-cloud damage assessment and DT synchronization 104

6.2 Proposed framework 104

6.3 Data preparation 105

6.4 Surface damage defection 106

6.4.1 Depth-to-grayscale transformation 106

6.4.2 DeepLabV3+ model 108

6.5 Spatial damage segmentation 112

6.5.1 Spatial voxelization 112

6.5.2 Binarization and morphological operation 113

6.6 DT synchronization and damage assessment 114

6.6.1 Data compression and transmission 114

6.6.2 Damage volumetric assessment 115

6.6.3 Model synchronization 116

6.7 Experiment preparation 118

6.8 Case study 1 – synthetic groove 119

6.8.1 Groove detection and spatial segmentation 119

6.8.2 DT synchronization 121

6.9 Case study 2 – real-world building crack 124

6.9.1 Crack detection and spatial segmentation 124

6.9.2 DT synchronization 126

6.10 Discussion 129

6.11 Summary 130

Chapter 7 Time-series IFD and health monitoring 134

7.1 IFD based on a uniaxial or triaxial sensor 134

7.1.1 Proposed IFD via AutoML-CNN and image-like fusion 134

7.1.2 Framework validation 139

7.1.3 Summary 146

7.2 Health-state recognition based on multiple sensors 147

7.2.1 Proposed multi-channel 1D-CNN architecture 147

7.2.2 Experiment validation 147

7.2.3 Result analysis 149

Chapter 8 BMKG data mining with text encoding and GraphSAGE 151
8.1 Theoretical foundation 151
8.1.1 Text encoding 151
8.1.2 GraphSAGE 153
8.2 Proposed approaches for maintenance knowledge mining 155
8.2.1 Knowledge-graph schema 155
8.2.2 Node classification 158
8.2.3 Link prediction 159
8.2.4 Loss function and evaluation metrics 160
8.3 Proof of concept 162
8.3.1 Experiment preparation 162
8.3.2 Node layer identification 163
8.3.3 Maintenance recommendation 166
8.4 Summary 169
Chapter 9 Bridge DT design and development 171
9.1 Overall design of system architecture 171
9.2 Preliminary platform and DT services 172
9.2.1 Digi-bridge visualization and CDE 172
9.2.2 Web-based API for damage segmentation 173
9.2.3 Unsupervised corrosion assessment 174
9.2.4 Question and answer chatbot 176
Chapter 10 Conclusion 178
10.1 Revisiting Research Questions 178
10.2 Research limitations and future work 182
10.3 Research contribution 184
References 187
Appendix A: Bridge inspection and repair actions – 15092 (primary data) 211
Appendix B: Templated chatbot for bridge maintenance query 213

List of Figures

Figure 1 Different DT system integration levels 2
Figure 2 Cloud-based Bridge DT Architecture 13
Figure 3 Triplet bridge-machine-human interaction 14
Figure 4 Defect localization with the drone control system 16
Figure 5 Capability of typical IoT wireless communication technologies 18
Figure 6 Crack detection by patch splitting and classification 21
Figure 7 Convex-hull fitting for damage quantification based on manual cutting 28
Figure 8 Compression through 3D run-length encoding 29
Figure 9 The IFD pipeline through traditional machine learning 30
Figure 10 IFD pipeline through deep learning 31
Figure 11 Reconstruction from time series to a single-channel grayscale image 33
Figure 12 The “research onion” 40
Figure 13 Overarching research roadmap 43
Figure 14 Complexity/ambiguity decreasing along the information hierarchy 47
Figure 15 A rooted binary tree for computing $f(x, y)$ through two-way communication 48
Figure 16 Bridge structural assessment based on a decision tree 50
Figure 17 Edge computing reduces time delay significantly when $Tcommunication$ dominates 50
Figure 18 (a) Cellular network based on star network (b) Non-cellular LPWAN based on the mesh network 52
Figure 19 Petri-net modelling for a bridge DT system initialized from $M0$ 56
Figure 20 The edge loop according to DIKW to reduce complexity 58
Figure 21 Proposed framework and developed prototype for bridge DT 61
Figure 22 MQTT implementation between cloud servers and HMI 62
Figure 23 Platform across Cesium (GIS part) and Xeokit (BIM part) 63
Figure 24 Training and test for crack detection 65
Figure 25 Crack segmentation through image processing 65
Figure 26 PPK approach for defect localization in the bridge coordinate system 66
Figure 27 Encoded defect information to synchronize for structural assessment 67
Figure 28 Synchronization of crack profile through LoRa and RLE 68
Figure 29 Developed bridge DT prototype for drone-enabled bridge inspection 69

Figure 30 1D-CNN architecture with multi-channel input utilized for pattern recognition 70
Figure 31 Test confusion matrices for SVM with signal features and SVM with WPE 71
Figure 32 Proposed framework and developed prototype for bridge VBM 72
Figure 33 (1) bridge network; (2) communication topology 73
Figure 34 (1) gateway-based route planning (2) sensor-node-based route planning 75
Figure 35 Proposed approach for few-shot damage detection 85
Figure 36 Decreased εest in constrained hypothesis space by prior knowledge 87
Figure 37 (1) $f_\theta(x_i)$ from supervised learning; (2) $f_\theta(x_i)$ from self-supervised learning 88
Figure 38 Entropy increases along with uncertainty rising in binary classification 90
Figure 39 Raw images (left) and hardcoded-transformed images (right) 91
Figure 40 1-shot and 5-shot crack identification with pre-trained ResNet18 (224×224) 92
Figure 41 2-way 5-shot performance of ResNet backbones in different depths (84×84) 93
Figure 42 Comparison of different pre-trained DNN embedding functions 95
Figure 43 Fine-tuning without entropy regularization 98
Figure 44 Fine-tuning with entropy regularization 98
Figure 45 Few-shot crack detection through the approach based on MAE 100
Figure 46 Few-shot spalling detection through the approach based on MAE 101
Figure 47 Proposed framework for 3D damage assessment and DT synchronization 105
Figure 48 Plane fitting and calibration of point cloud to check damaged column verticality 106
Figure 49 Pseudo grayscale images derived from point-cloud depth information 107
Figure 50 (a) training loss and validation loss; (b) training MIoU 109
Figure 51 Damage segmentation using DeepLabV3+ model on the pseudo grayscale images 111
Figure 52 Separated cuboid damage section and spatial voxelization 113
Figure 53 Voxelization through floor-like integerization 113

Figure 54 Surface binary mask and following morphological operations 113
Figure 55 3D Binary matrix compression and transmission 115
Figure 56 Comparison of convex-hull fitting and voxel-based fitting: (a) ground truth; (b) convex-hull fitting without manual separation; (c) voxel-based fitting 116
Figure 57 Designed FE model geometric updating through elementwise product 117
Figure 58 BIM model updating with the local damage geometry 118
Figure 59 (1) Synthetic grooves; (2) Real-world building crack 119
Figure 60 Groove detection on a pseudo grayscale image 120
Figure 61 Each layer (or slice) for the groove in the voxelized section 121
Figure 62 Groove volumetric assessment with convex-hull and voxel-based fitting 122
Figure 63 FE model geometric updating for synthetic groove 123
Figure 64 BIM model geometric updating for synthetic groove 124
Figure 65 Real-world crack detection with pseudo grayscale image via DeepLabV3+ 125
Figure 66 Each layer (or slice) for real-world cracking in the voxelized section 126
Figure 67 Volumetric assessment for the real-world building crack 127
Figure 68 FE model geometrically updating for real-world cracking 128
Figure 69 Proposed 3-channel pseudo-image reconstruction from time series 136
Figure 70 Proposed AutoML procedures for IFD 137
Figure 71 Proposed framework leveraging AutoML-CNN and image-like data fusion 139
Figure 72 CWRU bearing (a) and SEU gearbox (b) test rigs for data acquisition 140
Figure 73 IFD experiment for uniaxial acceleration data via the framework: (a) training loss; (b) validation accuracy 141
Figure 74 Test performance on the CWRU dataset through the proposed pipeline 142
Figure 75 (a) AUC for each model on the CWRU dataset and (b) confusion matrix of Xception on the CWRU dataset 142
Figure 76 IFD experiment for triaxial acceleration data via the framework: (a) training loss; (b) test accuracy 144
Figure 77 Test performance on the SEU dataset through the proposed pipeline 145
Figure 78 (a) AUC for each model on the SEU dataset and (2) confusion matrix of Xception on the SEU dataset 145
Figure 79 KW51 railway bridge, damage, and accelerator locations 148

Figure 80 Proposed 1D-CNN and test performance at data-level fusion...................149
Figure 81 Comparison of decision-level and data-level fusion on the test set..........150
Figure 82 Text encoding with BERT to generate node features 153
Figure 83 Proposed maintenance-oriented knowledge-graph schema for bridges..157
Figure 84 Node layer classification using DFS on intact and incomplete graphs.....158
Figure 85 Proposed pipeline for node layer classification by leveraging DFS, text encoding, and GraphSAGE 159
Figure 86 Proposed pipeline for link prediction by leveraging contrastive learning and GraphSAGE 160
Figure 87 Incomplete BMKG with 10%-35% missing links...163
Figure 88 Incomplete BMKGs after bidirectional DFS 164
Figure 89 Node classification for incomplete BMKG with text encoding through LLMs 166
Figure 90 Initial AUC and post-processed AUC for link prediction.............................167
Figure 91 Contextural link prediction based on semantic similarity 168
Figure 92 Extra predicted links for incomplete BMKG with 35% missing links 168
Figure 93 Designed workflow for AI-assistive bridge maintenance by leveraging NLP and GNNs..169
Figure 94 The overall architecture of the bridge DT system 171
Figure 95 Digi-bridge visualization and CDE...173
Figure 96 Test performance of the trained DeepLabV3+ models...............................173
Figure 97 Web service architecture for damage segmentation 174
Figure 98 Web service for damage segmentation via DeepLabV3+ 174
Figure 99 Corrosion segmentation via k-means in different colour spaces...............175
Figure 100 Corrosion assessment via k-means in L*a*b..176
Figure 101 BMKG based on Neo4j 176

List of abbreviations

AEC = architecture, engineering, and construction

AI = artificial intelligence

AutoML = automated machine learning

BMKG = bridge maintenance knowledge graph

BMS = bridge management system

BIM = building information modelling

BrIM = bridge information modelling

CNN = convolutional neural network

DL = deep learning

DNN = deep neural network

DT = digital twin

FCN = fully connected network

FSL = few-shot learning

GIS = geographic information system

GNN = graph neural network

HMI = human-machine interface

IFC = Industry Foundation Classes

IFD = intelligent fault diagnosis

IoT = Internet of Things

KG = knowledge graph

KE = knowledge engineering

LPWAN = low-power wide area network

ML = machine learning

O&M = operation and maintenance

PE = physical entity

ProtoNet = prototypical network

Q&A = question and answer

RUL = remaining useful life

SHM = structural health monitoring

TLS = terrestrial laser scanning

tinyML = tiny machine learning

VE = virtual entity

List of outcomes

Journal Papers:

1. **Gao, Y**., Li, H., Xiong, G. and Song, H. 2023. *AIoT-informed digital twin communication for bridge maintenance*. *Automation in Construction* 150, article number: 104835. (10.1016/j.autcon.2023.104835) **(Impact factor: 10.3)**

2. **Gao, Y.**, Li, H. and Fu, W. 2023. *Few-shot learning for image-based bridge damage detection*. *Engineering Applications of Artificial Intelligence* 126(PartC), article number: 107078. (10.1016/j.engappai.2023.107078) **(Impact factor: 8.0)**

3. **Gao, Y.**, Chai, C., Li, H. and Fu, W. 2023. *A Deep Learning Framework for Intelligent Fault Diagnosis Using AutoML-CNN and Image-like Data Fusion.* Machines 2023, 11, 932. (doi.org/10.3390/machines11100932) (**Impact factor: 2.8**) **(Vol. 11, Iss. 10, October 2023, Cover Story)**

4. **Gao, Y.**, Li, H., Fu, W. and Chai, C. *Damage Volumetric Assessment and Digital Twin Synchronization Based on LiDAR Point Clouds. Automation in Construction 157*, article number: 105168. (10.1016/j.autcon.2023.105168) **(Impact factor: 10.3)**

5. **Gao, Y.**, Li, H., Richards, J., Xiong, G. and Fu, W. *Exploring Bridge Maintenance Knowledge Graph by Leveraging Text Encoding and GraphSAGE*. Submitted to *Automation in Construction*. (**Under review**)

Conference Papers:

1. **Gao, Y.**, Li, H. and Xiong, G. 2022. *An efficient and resilient digital-twin communication framework for smart bridge structural survey and maintenance* (doi.org/10.7146/aul.455.c207). Presented at *the 29th EG-ICE International Workshop on Intelligent Computing in Engineering*, Aarhus, Denmark, 06-08 July 2022.

2. **Gao, Y.**, Li, H., Fu, W. and Xiong, G. 2023. *Few-shot classification for image-based crack detection*. Presented at the *30th EG-ICE: International Conference on Intelligent Computing in Engineering*, London, UK, 4-7 July 2023.

Chapter 1 Introduction

1.1 Research background

In recent years, the construction industry has witnessed a transformation towards the digitalization of assets, which is not only a trend but also a substantial evolution in conceiving, managing, and interacting with built environment. Digital assets, embracing sensor data from Internet of Things (IoT) devices, building information, 3D models, and extensive historical maintenance records, have become increasingly important. Different from traditional physical assets, these digital counterparts provide a dynamic, interactive, and predictive approach for project management and infrastructure maintenance. Currently, digital assets in construction have transitioned from supplementary tools to core elements that drive value throughout projects. They play a crucial role in enhancing decision-making processes, optimizing operation and maintenance (O&M) costs, elevating safety criteria, and extending the lifecycle of infrastructures. This transformation is characterized by an increasing awareness in the industry that digital assets hold inherent value and can yield substantial enhancements in project outcomes and operational efficiencies.

Digital twinning is an emerging technology for intelligent asset management that provides an up-to-date presentation of an actual physical asset in operation, which can reflect the current asset condition, include relevant historical information, and make holistic decisions (or feedback). Although the idea can date back to 1970 as the “living model” of the Apollo 13 space rescue mission, the first definition of a concept nowadays known as the digital twin (DT) was made in 2002 by Michael Grieves in the context of an industry presentation concerning product lifecycle management (PLM) [1]. Based on the given definitions of a DT in many studies [2]–[7], there are two common understandings of the term DT. One can be identified as the digital counterpart of a physical object or entity (PE); the other refers to the DT systems or frameworks. To prevent misunderstandings, in this research, DT refers explicitly to the latter, i.e., a DT system, and the former (i.e., digital counterparts) is referred to as a virtual entity (VE). Moreover, the DTs can be categorized in terms of the system integration level between PEs and VEs [5], i.e., digital modelling, digital shadowing, and digital twining, as shown in Figure 1, where the dashed line represents the manual data flow, and the full line indicates the automatic data flow.

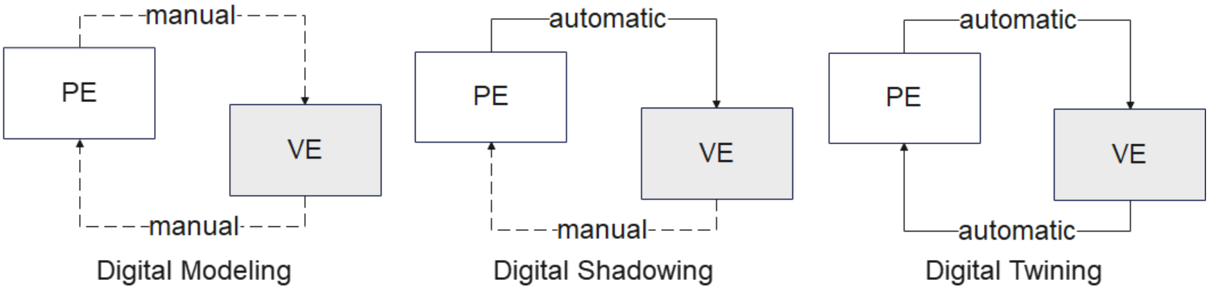


Figure 1 Different DT system integration levels [5]

A qualified DT for bridge O&M is not only about creating a digital representation of the physical bridge (i.e., digital modelling) but can also update in near real-time as new data is collected (i.e., digital shadowing) and provide feedback into the physical bridge based on relevant historical records as well as perform 'what-if' scenario analysis for asset risk assessment and performance prediction (i.e., digital twining) [8]. In practice, bridge DT is systematic engineering, consisting of inspection and monitoring on the site of physical bridges, communication between physical entities (PEs) and virtual entities (VEs), and backend services, including analysis, simulation, prediction, and decision-making.

Many efforts have been made for the related topics of the bridge DT in previous research, such as drone-enabled bridge inspection [9], [10], real-time IoT-based bridge structural health monitoring (SHM) [8], bridge information modelling (BrIM) (served as DT informatics foundation) [11], [12], knowledge-based and optimization-oriented bridge management systems (BMS) [13], [14], etc. Nevertheless, there is limited research exploring the entire maintenance cycle of DTs for bridges. This includes stages from on-site inspection and monitoring, synchronization of DT, and feedback to

Physical Entities (PEs), to the backend services that facilitate decision-making in the bridge O&M.

1.2 Research motivations

Although DT has garnered significant interest for its potential to revolutionize bridge O&M, many gaps exist between academic research and industrial applications. For example, when DT implementation concerns bridge locations, it may have many issues, such as restricted communication capability and power supply, which is a bottleneck for DT synchronization with massive inspection and monitoring data; moreover, the automatic visual inspection based on supervised learning requires a huge number of annotated images, but it is not always practical to collect sufficient examples for the model training due to various materials, different surface coatings, changing light and weather conditions, etc.; furthermore, assessing bridge conditions and proposing repair actions for bridge maintenance, which are based on the specific expertise, are still time-consuming and require elaborate manual analysis by engineers in routine practice. Nevertheless, there is limited research examining the efficiency and resilience of bridge DTs and associated methods, which could improve their practical application in bridge maintenance. Hence, the first motivation of this research is to fill the gaps between the conceptual DTs and the practical applications in bridge O&M.

Another research motivation is to promote the existing bridge SHM and BMS systems to bridge DTs in terms of both integration and back-end services for bridge O&M. For example, the traditional bridge SHM systems are often reactive and focused on condition monitoring by using alarms to indicate damage happening [8] but hardly provide proactive warning or remaining useful life (RUL) estimation of the property according to future ambient change and traffic loads for preventative maintenance, as well as optimized operation planning in an area. The latter requires simulation and prediction. Moreover, it is also of great significance to integrate the mainstream inspection, monitoring, assessment, and decision-making technologies (which were stand-alone and manual) into the same automatic digital twining pipeline, thereby achieving an efficient, effective, and comprehensive DT for the bridge maintenance cycle. For example, the TLS survey is usually project-based and time-consuming due to its massive data and the complexities involved. Integrating the real-time (or near real-time) TLS into a DT for bridge O&M does not only enhance the damage assessment accuracy at the back end by offering highly detailed 3D information but can also improve the inspection effectiveness on the site by leveraging the feedback from the DT such as historical records and maintenance expertise guiding inspector to areas of concerns.

1.3 Research questions

Following the above background and motivations, this research aims to solve the problems encountered by DTs for practical applications in the complete bridge maintenance cycle, mainly focused on efficiency, effectiveness, and intelligence. This research objective can be broken down into the following questions:

1) How can an efficient, resilient, and bidirectional DT framework be created for practical bridge O&M in challenging conditions with massive heterogeneous data and limited communication? – **Chapter 4**

2) How can automatic bridge image-based defect detection be achieved under complex scenarios with weakly supervised information, such as very limited data? – **Chapter 5**

3) How can the dense 3D point cloud data (such as from TLS) be integrated into the digital twinning pipeline efficiently, achieving DT model synchronization and volumetric assessment for local damages in near real-time? – **Chapter 6**

4) How can intelligent fault diagnosis (IFD) and health-state monitoring be implemented through automated machine learning (AutoML) and data fusion

(multi-axial or multi-sensor) based on time series? – **Chapter 7**

5) How can we design a bridge maintenance knowledge graph (BMKG) based on the bridge structure and the practical inspection reports and then achieve maintenance recommendation via the analogical strategy? – **Chapter 8**

1.4 Research innovations

This research contains works on theoretical and practical developments for DT implementation in the bridge maintenance cycle. The innovations refer to the overarching DT framework, the on-site inspection and monitoring, the communication and model synchronization, as well as backend services for assessment and decision-making, which can be listed below:

1) The time delay of a DT between PE and VE is studied theoretically and mathematically, which is significant to the design of a bridge DT for different purposes. Then, an AIoT-informed bridge DT framework is proposed by leveraging low-power wide-area network (LPWAN) to achieve a decentralized fault-tolerant DT system and by leveraging edge computing to minimize communication complexity.

2) A training-free few-shot learning (FSL) approach based on an improved prototypical network (ProtoNet) is proposed for image-based bridge damage detection with weakly supervised information, which can be quickly implemented under complex circumstances with very limited data.

3) An extremely light-weight framework is proposed to integrate dense point cloud data (PCD) into the digital twining during a survey, which can achieve 3D spatial damage segmentation via voxelization, low-latency model synchronization through a binary matrix and lossless compression, as well as more accurate volumetric assessment without manual efforts.

4) An efficient framework leveraging AutoML and data fusion (multi-axial or multi-

sensor) is proposed for IFD and health-state monitoring based on time series signals.

5) A BMKG schema is designed based on bridge structures and practical inspection reports. Then, an approach that can automatically suggest repair actions for new defects is proposed by leveraging the analogical strategy based on context.
6) A preliminary web-based bridge DT platform fusing GIS, BIM, and AI is developed, integrating element-based query, visualization, vision-based bridge defect assessment, and a knowledge-based question and answer (Q&A) chatbot.

The above research outcomes significantly advance the use of digital assets in construction, particularly in bridge maintenance. By developing a sophisticated AIoT-informed DT framework, the research enables bridge stakeholders to make more accurate and efficient decisions through data synchronization and analysis in near real-time. The implementation of training-free few-shot approach for damage detection is promising to revolutionize automatic visual inspection, allowing for rapid, effective analysis with minimal data. The integration of dense PCD into digital twinning and the use of AutoML based on time series greatly enhance efficiency and precision for reality modelling and bridge SHM during routine inspection and monitoring. Additionally, the creation of a maintenance-oriented KG schema and a web-based DT platform facilitates easier management and maintenance planning of digital assets. These innovations not only streamline bridge O&M processes but also exemplify the broader potential of digital assets in enhancing operational efficiency, reducing costs, and improving the lifecycle management of many other construction projects.

1.5 Structure of the thesis

Chapter 1 aims to comprehensively introduce the research’s background, motivation, and significance within this thesis.

Chapter 2 provides a thorough review of the existing body of literature. It is divided into five main sections: (1) the existing bridge DT frameworks and systems; (2) the details of various bridge maintenance technologies within the context of triplet interaction between bridge, machine, and human; (3) the details of vision-based bridge inspection technologies, including image-based and point-cloud-based survey; (4) the workflows for IFD and health-state monitoring based on traditional machine learning (ML) and deep learning (DL), respectively; (5) the advanced bridge knowledge engineering (KE), including ontology generation and KG completion, as well as data mining using GNN.

Chapter 3 presents the research methodology in the form of a “Research Onion” from the outer layers to the inner layers, including philosophies, approaches, strategies, choices, time horizons, techniques and procedures. Then, the research roadmap is proposed with the methodology details, hypotheses, and expected outputs.

In **Chapter 4**, an efficient and fault-tolerant AIoT-informed DT framework is proposed for bridge maintenance, which can decrease time delay, minimize communication complexity, and enhance system resilience against fault. The validation is conducted using a DT prototype and simulation on three different bridge O&M scenarios: drone-assisted bridge inspection, vibration-based monitoring (VBM), and dynamic evaluation, demonstrating the framework’s superiority compared to the existing bridge DT frameworks and systems.

Chapter 5 focuses on image-based bridge damage detection with weakly supervised information using FSL approaches. The chapter begins with the theoretical foundation for few-shot classification (FSC), including problem definition, feature embedding, transfer learning, and fine-tuning. Then, it proposes an improved ProtoNet for few-shot bridge damage detection based on ablation studies. Finally, the network's validation uses real-world bridge defect images, demonstrating its fast implementation capability.

Chapter 6 proposes an efficient voxel-based framework for integrating the dense point-cloud data from TLS into a digital twinning process. The framework consists of three modules: surface damage detection, spatial damage segmentation, and DT synchronization, which can enable model synchronization and volumetric assessment for local damages in near real-time. The validation is conducted using synthetic grooves and real-world building cracks, demonstrating its efficiency, effectiveness, and superiority compared to the previous research.

Chapter 7 delves into the time-series-based IFD and SHM. Firstly, a data-driven framework is proposed for edge-based IFD integrating image-like data fusion, AutoML-CNN, and tiny machine learning (tinyML). Its validation uses two public IFD datasets based on uniaxial and triaxial sensors. Secondly, a 1D-CNN architecture is employed for edge-based bridge SHM by leveraging multi-sensor fusion. The validation is performed using data from a VBM project for a railway bridge. Meanwhile, the data-level and the decision-level fusions are compared in the experiment.

Chapter 8 explores the knowledge mining of a BMKG using natural language processing (NLP) and GNN for semantic enrichment and maintenance recommendation. The proposed methods involve contextual node classification and link prediction by leveraging GraphSAGE and contrastive learning. The validation uses a BMKG derived from an actual bridge inspection report. Finally, an intact workflow is designed for practical implementation.

Chapter 9 presents the developed bridge DT system, including the overarching design, the preliminary web-based platform, and a range of backend services dedicated to bridge O&M.

Chapter 10 concludes this thesis by revisiting and answering the research questions. In this section, the entire research is discussed and reflected. The limitations and future work are clarified. Additionally, the major contributions of this research are summarized.

Chapter 2 Literature review

In order to produce a comprehensive review of the subject, this chapter is divided into five sections: (1) Section 2.1 introduces the DT development and characteristics, as well as the existing bridge DT frameworks and systems; (2) Section 2.2 elaborates the technologies associated with bridge DTs in the context of the interaction between bridge, machine, and human; (3) Section 2.3 reviews the current situation of vision-based bridge inspection, including via images and point clouds, as well as the issues encountered in practice; (4) Section 2.4 reviews the IFD approaches with time series signals, which can be integrated into the bridge DT for health-state monitoring; (5) Section 2.5 reviews the current studies in KE associated with bridges and data mining with GNN, and then introduces the potential to improve bridge maintenance by leveraging BMKG, GNN, and contextual analogy.

2.1 Bridge DT frameworks and systems

In the context of Industry 4.0, the architecture, engineering, and construction (AEC) processes and product life-cycle management are becoming more efficient and intelligent [15], and bridge O&M is no exception. As an emerging technology, DT has received extensive attention and application in different fields for this revolution [16], especially when combined with digital transformation, such as DT-enabled smart manufacturing, construction, operation, and management. Although there is hardly a unified framework for DTs across different disciplines and domains, the ongoing research exhibits a notable characteristic (or pattern) in the current studies that DT systems are always designed and developed for specific purposes and circumstances [17], [18].

In terms of bridge engineering, a bridge DT for SHM can be defined as a virtual representation of the physical bridge, which does not only update as new data is collected in near real-time but also provides feedback into the physical bridge and performs ‘what-if’ scenarios for assessing asset risks and predicting asset performance [8]. A DT for bridge maintenance aims to be updated along with visual inspection and non-destructive test (NDT) continuously [19]–[21] and integrated with other multi-source data, such as original design, damage history, inventory, traffic, weather, disaster, to support holistic decision-making for maintenance planning [22], [23]. Moreover, multiple bridges DTs can be considered a bridge network and utilized for intelligent transport, usually represented topologically on a map [24]. A comprehensive and sophisticated DT system to support bridge O&M in this research aims to consider all the above purposes (or services). The bridge DT models can be created through different approaches, including building information modelling (BIM), physics-based approach (such as finite element modelling), data-driven approach (such as statistical modelling), and data-centric engineering approach (i.e., hybrid modelling) [8]. Their primary characteristics comprise digital replica (like geometry, materials, etc.), data composition, bidirectional connection for updates and feedback in near real-time, the entire life-cycle coverage of a physical bridge, a common data environment (CDE), visualization capabilities, simulation, and the ability to learn from actual measurement data [8].

Over the past decade, the interest in bridge DT has grown significantly, and a few prototypes and pilot projects have been proposed and demonstrated successfully. For example, a DT system for two pilot railway bridges [25] was developed, which integrates Fiber Bragg Gratings (FBG) sensors, laser rangefinders, and other additional sensors to achieve ongoing monitoring for train-bridge coupling parameters (e.g., strains, accelerations, train axle positions). It can calculate the key indicators (e.g., curvature, end rotations, displacements, axle weights) for SHM in the cloud and use a web-based platform embedded with Unity for human-machine interaction (HMI). Another cloud-based bridge DT for SHM [26] employs a finite element model with damaged states and data synthetically created, real-world monitoring data (such as vibration and strain) from multiple sensors, and a pre-trained surrogate model based on DL to detect damage existence, identify damage location, and quantify damage severity on a practical bridge, thereby achieving proactive maintenance. Moreover, a conceptual bridge DT [21], [27], [28] for preventative maintenance is developed using a surface model, BIM model, and simulation model. The surface model is generated continuously through reverse engineering (photograph mapping and 3D scanning) and aligned parametric modelling during the bridge lifecycle O&M. Damage information through image processing after visual inspection can be recorded in a code system and linked to specific BIM elements. The simulation model is achieved with FEM, wherein the detected deterioration of structural elements is evaluated and employed to update the structural parameters. Another exemplary openBIM-based bridge DT [29] was developed by industry, which can enable long-term monitoring of bridge condition and predictive maintenance (PM) with aggregated information by combining traditional inspections with digital information (from structural diagnosis and monitoring) and injecting the derived semantic information into the BIM model. Furthermore, there are also many other frameworks or systems developed in the forms of the bridge BMS or BrIM for bridge SHM [30]–[34], and O&M [23], [35]–[37]. They are very instructive due to the close relationship to bridge DT, which can be taken as resorts and further developed to bridge DT.

Although there are already some successful designs and implementations for bridge DT, most of them relied on excellent communication [25], [29], [38], such as Ethernet, 4G, and 5G, or even did not reveal communication approaches such as conceptual designs [21], [27], [28]. However, when the bridge DT is implemented in practice, the location may be a concern because many bridges work with limited communication and power supply, which can be attributed to geographic or economic reasons. For example, the successful bridge DT exemplar in the pilot project [25], which requires excellent communication for successful DT implementation, cannot spread widely to thousands of bridges in the UK because it will lead to significant expenditures on middleware and data plans. Meanwhile, storing such tremendous inspection and monitoring data will also become a heavy burden for technology and the economy.

Besides, as shown in Figure 2, most of the existing bridge DTs work in a centralized mode based on the cloud [37]–[40]. In this way, the digital bridge can synchronize with the physical bridge through the triplet bridge-machine-human interaction, which involves four significant components, i.e., data acquisition and preprocessing, communication, cloud servers, human-machine interface (HMI), to achieve multiple DT services at the back end.

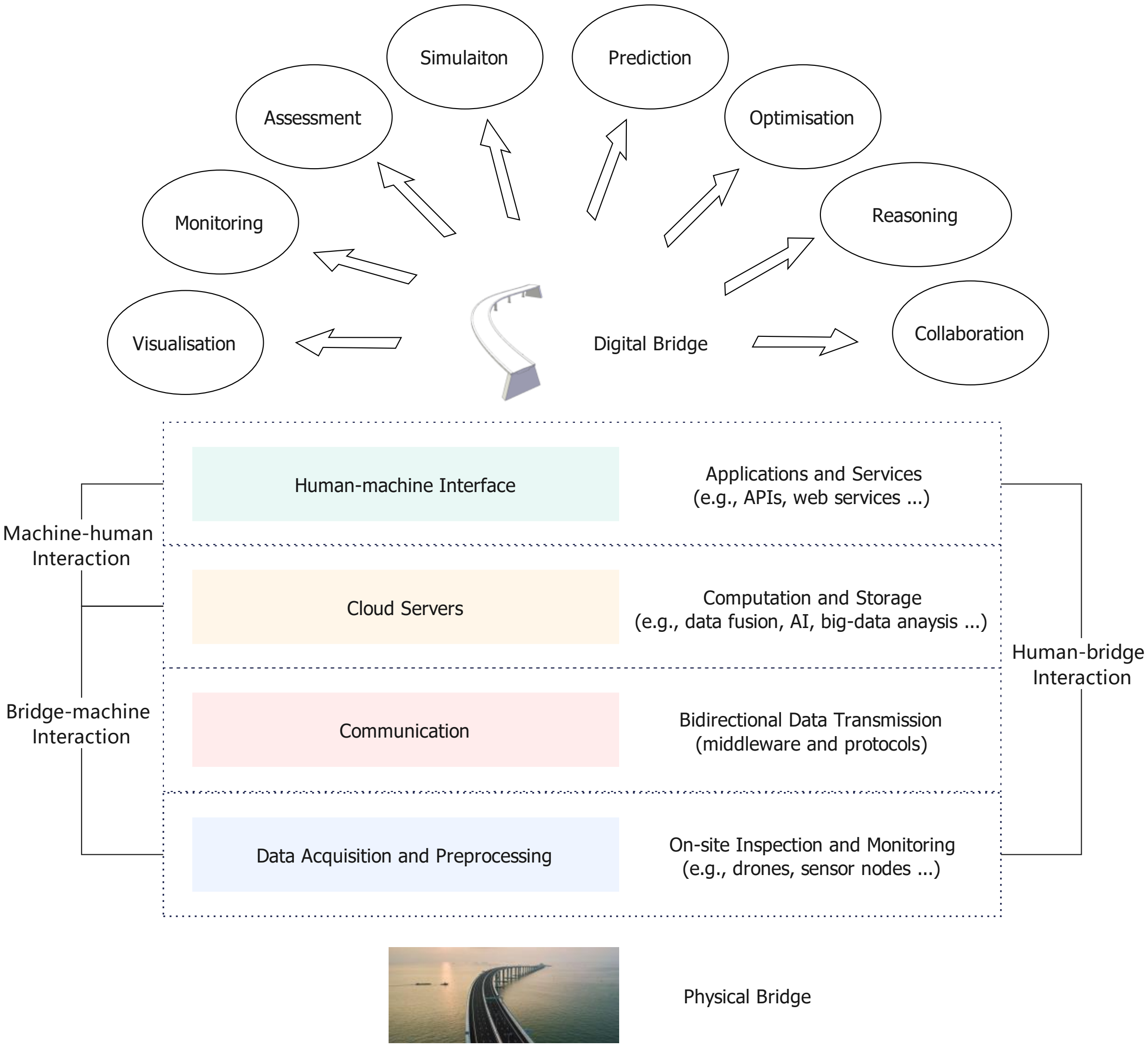

Simulaiton
Prediction
Assessment
Optimisation
Monitoring
Reasoning
Visualisation
Digital Bridge
Collaboration
Human-machine Interface
Applications and Services
(e.g., APIs, web services ...)
Machine-human Interaction
Cloud Servers
Computation and Storage
(e.g., data fusion, AI, big-data anaysis ...)
Human-bridge Interaction
Bridge-machine Interaction
Communication
Bidirectional Data Transmission
(middleware and protocols)
Data Acquisition and Preprocessing
On-site Inspection and Monitoring
(e.g., drones, sensor nodes ...)
Physical Bridge

Figure 2 Cloud-based Bridge DT Architecture

However, the above centralised cloud-based architecture raises two questions. Firstly, do we need to transmit such massive heterogeneous data for bridge DT, especially in a communication-constraint environment? Secondly, the issue of resilience in bridge

DTs [39], [40] is usually neglected, so how can we enable bridge DT with fault-tolerant capability, e.g., under a temporary loss of communication?

According to the investigation of the UK practitioners [41], the first question is related to the significant disconnect between academia and industries towards DT applications in bridge O&M, which is the difficulty of updating the bridge DT models automatically in routine practice. It can be solved by collecting data continuously from traditional inspections and ongoing structural monitoring, but one of the bottlenecks is focused on the synchronization of massive heterogeneous data. Besides, to our best knowledge, there is no precedent of bridge DT framework considering resilience to endure a temporary loss of communication. Hence, this work is going to study the bridge DT communication to solve the above issues from two perspectives: 1) to enhance DT efficiency through a trade-off between complexity and time delay of DT services; 2) to enable bridge DT with excellent fault tolerance, which can endure a temporary loss of communication.

2.2 Bridge-machine-human triplet interaction

This part investigates the technologies related to bridge DT, which is helpful for the following framework design and prototype development. The content will be organized as a triplet bridge-machine-human interaction, as shown in Figure 3.

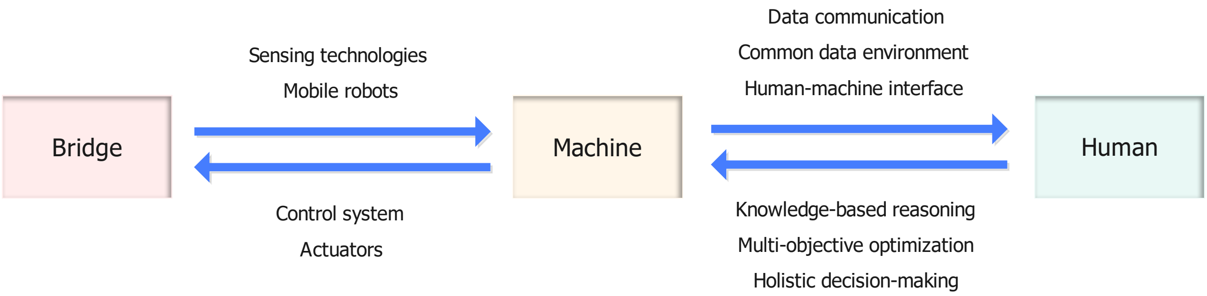


Figure 3 Triplet bridge-machine-human interaction

2.2.1 Bridge-machine Interaction

2.2.1.1 IoT-based monitoring

Bridge-machine interaction includes various sensing and robotic technologies. Recently, elaborated attached sensors or handheld equipment have been successfully applied on bridges [42] for global SHM, including accelerometers [30], strain gauges [43], fibre optical sensors [34], etc., and local non-destructive testing (NDT), such as passive acoustic emission (AE) monitoring [44], active sonar [45] and ultrasonic coda wave interferometry (CWI) detection [46]. A high-speed motion camera can also monitor bridge vibration and displacements with high accuracy under dynamic loadings based on motion amplification technology [32]. Digital single-lens reflex (DSLR) [47], [48] and infrared (IR) cameras [47], [48] are also widely used for vision-based bridge inspection to identify surface deficiencies due to their easy integration into traditional inspection workflows. LiDAR (i.e., 3D laser scanning) is another powerful vision-based method supplementary to 2D inspection [49] for bridge inspection, which can provide more detailed information such as depth. These sensors and equipment in the regular inspection and real-time monitoring will generate massive heterogeneous data, which is essential for bridge condition assessment and decision-making but also poses a challenge for DT synchronization.

2.2.1.2 Robotic inspection

Traditional bridge inspection with manual access is costly, time-consuming, and dangerous [50]–[52]. Nowadays, it has got a lot of help from versatile robotic systems, e.g., drones [50]–[52] and climbing robots [50]–[52]. Such automated inspection systems can work with various payloads (i.e., sensors) to access limited areas and better angles that are difficult or dangerous for people to reach. Moreover, they are not only carriers of inspection payloads but can also improve or assist the inspection with the data from their control system. For example, mobile robots and drones can use the approaches based on Global Navigation Satellite System (GNSS) [53] – i.e., real-time kinematic (RTK) and post-processing kinematic (PPK) positioning, obstacle avoidance system – camera, ultrasonic distance ranger or LiDAR, inertial measurement units (IMUs) to help with defect localization, as shown in Figure 4. Thus, it will further increase the amount of data for synchronization.

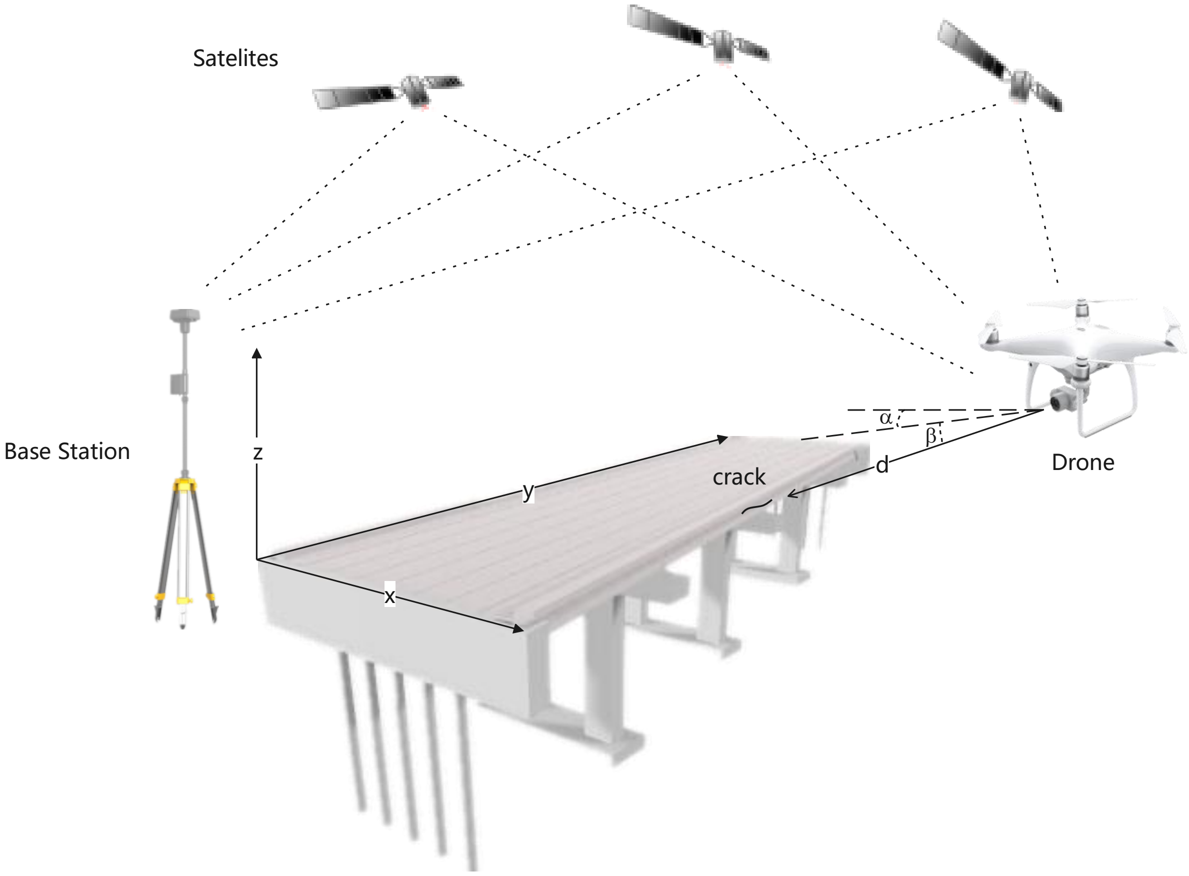


Figure 4 Defect localization with the drone control system [54]

2.2.2 Machine-human Interaction

2.2.2.1 Data communication

Machine-human interaction includes data communication, storage, common data environment (CDE), and HMI. Transmitting such massive data from bridge-machine interaction requires abundant bandwidth. Therefore, data acquisition in many DTs is based on a wired connection, such as Fieldbus and Ethernet, which can provide fast (up to 10Gbps) and robust data transmission. Still, it has low scalability and leads to high installation and maintenance costs [55]. Wireless communication is more flexible, enabling monitoring of the bridges that used to be inaccessible by cables, although it is susceptible to distance and obstacles. The capability of different wireless communication technologies [56]–[58] is shown in Figure 5. Short-range wireless communication, such as WIFI, Zigbee and Bluetooth, is only suitable for data collection in situ. Commercial cellular networks operate at a medium range with service costs (charge of data plan). Their bandwidth increases along with frequency bands, i.e., data rate – 5G > 4G > 3G, but their coverage decreases, i.e., distance – 3G > 4G > 5G. The LPWAN technology, which is long-distance wireless communication, includes cellular and non-cellular. Cellular LPWAN, such as NB-IoT and LTE-M, relies on existing commercial cellular networks. In contrast, non-cellular LPWAN, including LoRa, Sigfox, Ingenu, etc., works on free, unlicensed industrial, scientific, and medical (ISM) bands. LPWAN is a restricted communication, which works with limited bandwidths (i.e., low data rates), e.g., LoRa (sub-GHz) – up to 50kbps, NB-IoT – up to 158.5kbps, GBAN – up to 800kbps, LTE-M – up to 1Mbps, and is usually bounded by small payload size and duty cycle, such as LoRa and Sigfox. Still, it has many advantages, such as long-distance, scalable, low-cost, suitable for bridges in resource-constraint environments and remote areas and promising for widespread applications.

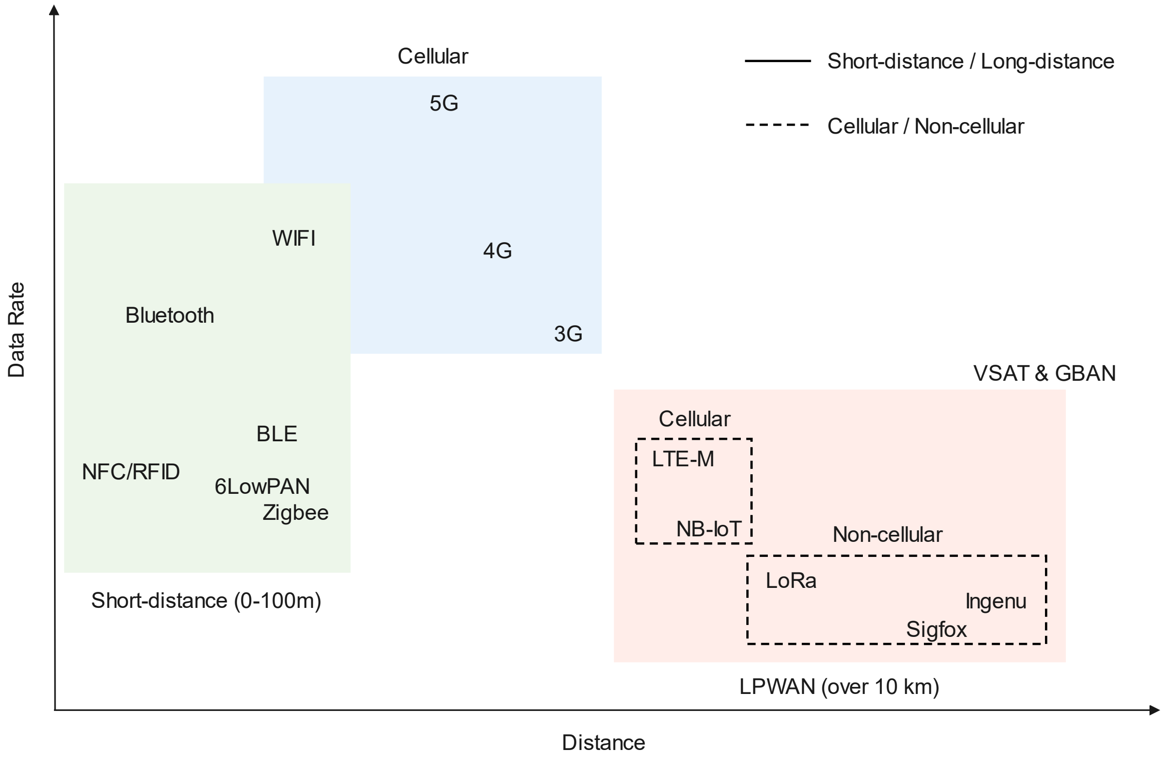


Figure 5 Capability of typical IoT wireless communication technologies

2.2.2.2 Common data environment

A CDE is required for bridge DT to store massive heterogeneous data from multiple data sources to support seamless collaboration among different stakeholders across the bridge life cycle, including monitoring data, inspection reports, bridge design and construction documents, historical records, rules and standards, inventory, as well as ambient data, such as traffic, weather, air salinity, water speed, and natural disasters. Users can conditionally access and modify the models and data [8]. Meanwhile, each user should have a corresponding priority and resource budget, including time and payment, to avoid potential conflicts with others [43]. It is also worth noting that the required computation and storage resources should not exceed the total capacity of the service provider [43]. BIM with Industry Foundation Classes (IFC) format files is the most popular choice to create such a CDE [59], [60] thanks to the consistent and sharable data schema. For example, a dynamic data-driven environment [61] based on IoT-informed BrIM was developed for bridge SHM to support dynamic visualization, seamless updating, long-term monitoring, and data exchange with IFC. Another framework based on BrIM for drone inspection was proposed for data storage and management, which can assign deficiency evaluation to the corresponding BIM element [62].

2.2.2.3 Bridge DT platform

A virtual platform in computer terminals integrating multiple tools and applications is required to support human-machine interaction [61], [63]–[65]. A few tools have been successfully validated for bridge DT visualization, including Unity [65], Xeokit [66]–[68], and Visualization Toolkit (VTK) [69]. Docker is recommended for platform deployment as it is less resource-intensive (i.e., multiple containers can share a common kernel) than virtual machines (VMs), and the services can be distributed amongst a cluster of nodes via Docker Swarm [70]. The Hadoop framework can be used for big-data storage. Meanwhile, NoSQL (such as MongoDB) and NewSQL databases with MapReduce and Spark can accelerate computation for data management [71]. Moreover, the web-based application relying on microservices [33] has recently been taken as an alternative to a desktop-based application due to its lightweight and excellent cross-platform capability. Additionally, extended reality and gaming technologies, such as augmented reality (AR), virtual reality (VR), mixed reality (MR), and Unity engine, are brought in to develop new HMI [72]–[74]. Emails, text messages, as well as social media (such as Twitter and Facebook) can also be adopted in the platform [75], [76].

2.2.3 Human-bridge Interaction

Human-bridge interaction aims to make holistic decision-making based on human knowledge or human-designed computational programs. It is reflected in the various DT functions (or services) for bridge O&M, such as condition assessment, maintenance planning, and multi-objective optimization. Traditionally, bridge structural integrity and serviceability are assessed by human engineers based on experience and standards [77]. Currently, many data-driven and knowledge-driven approaches are employed in the workflow, such as data analysis, data mining, ML, and knowledge discovery. The data usage in this process can be either indicator-based [78] or the direct use of data in the time or frequency domain [79]. For example, structural damage diagnosis can be achieved through a fully convolutional encoder-decoder architecture using vibration signals from a grid sensor network to locate damage and classify multiple damage mechanisms [80].

Moreover, bridge deterioration can be predicted based on data mining and knowledge discovery from multi-source data, such as design and construction documents, inspection reports, traffic, weather, disasters, and inventory [23]. Meanwhile, the intervention duration and impact caused by maintenance can also be predicted through a deep neural network (DNN) with embedded entities [81], which can help to make optimal maintenance planning. Additionally, early warnings and protective measures, such as weight restriction, traffic diversion, or even closure, are significant for physical bridges' safety and on-site users' safety, especially in an emergency. Such decision-making is also required in the bridge DT, which is usually achieved by knowledge-based reasoning, such as ontology [82] or knowledge graph [35], [36], [82]. The human-bridge interaction can also enhance maintenance efficiency and save social and economic costs.

2.3 Vision-based bridge inspection

2.3.1 2D image-based survey

2.3.1.1 Damage classification and detection

For bridge visual inspection, a fundamental task is to determine if there are certain kinds of damage in an image, such as surface cracking, spalling, or rebar corrosion, i.e., damage classification [83]. The task can be defined as the binary classification for each defect or a multi-defect classification. It can also be extended to determine whether damages exist and deduce the exact damage type, such as longitudinal crack, transverse crack, and alligator crack [83]. Furthermore, damage detection aims to provide more information about the damage, such as location, area, skeleton, and direction, which is helpful because classification only indicates the existence of defects in an image but leaves the task of finding the actual defect to inspectors [83]. A typical damage detection approach can be achieved by sliding the window or splitting the image into patches and then applying classification on each window or patch, followed by stitching them back, as shown in Figure 6. Another type of damage detection utilizes bounding boxes to indicate defects, like object detection tasks in many competition datasets, such as COCO [84] and Pascal VOC [85]. However, this method is not always the best option to locate damage because defects have various shapes. The created bounding box can include many undefective sub-regions, e.g., an oblique crack is marked by a sizeable bounding box determined by its diagonal points.

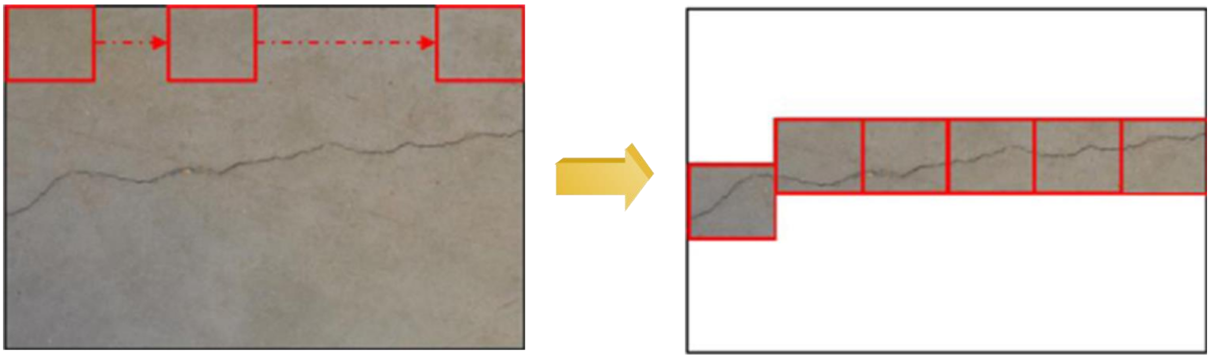


Figure 6 Crack detection by patch splitting and classification [86]

The image-processing methods for damage detection underperform on practical inspection images due to the interference of surface textures, changing light, stains, etc. [87]. Therefore, many data-driven approaches have been developed based on artificial intelligence (AI) for damage classification and detection to assist visual inspection. They can be categorized based on feature extraction, i.e., traditional ML (with handcrafted features) and DL (without handcrafted features). The former include support vector machine (SVM) [88]–[90], Random Forest [91]–[94], Adaptive boosting (Adaboost) [95], [96], artificial neural network (ANN) [97]–[100], etc. In traditional ML-based approaches, image processing is still required to implement pre-defined feature extraction. Various features have been utilized in research, such as statistical information, feature map projection, and defined defects' characteristics [101]. For example, Chen et al. [102] utilized local binary patterns (LBP), SVM, and Bayesian decision theory to detect cracks; Wang et al. [103] employed crack characteristics (i.e., density and connectivity) and SVM to discriminate alligator and transverse cracking. Meanwhile, ML can also be used to find optimal parameters for feature extraction, such as threshold values [100], [104]. The major problem with traditional ML approaches is that they still require handcrafted features and contain shallow learned information (or representation) [101].

Deep learning (DL) can extract features automatically with multi-layer neural networks. Cha et al. [86] proposed a convolutional neural network (CNN) to identify cracks without calculating handcrafted features for the first time. The model was trained on 40k images (256 ×256), including crack and non-crack, and then combined with the sliding window to scan any photo larger than 256 ×256 for crack detection, which shows better performance and can detect concrete cracks in practical scenarios. Subsequently, a few datasets and DL approaches were created for damage detection based on supervised learning, including CNN [87], [105]–[107], transformer [108]–[112], etc. For example, Xu et al. [107] created an image set for automatic bridge crack detection. They proposed a CNN architecture by leveraging the atrous spatial pyramid pooling (ASPP) module and depth-wise separable convolution, which can achieve 96.37% accuracy on the test set. Xiang et al. [112] integrated a transformer module in YOLOv5 for road crack detection. Cha et al. [113] created a dataset including five typical defects – concrete rack, steel corrosion with two levels (medium and high), bolt corrosion, and steel delamination. Then, they employed the faster region-based convolutional neural network (Faster-RCNN) and the region proposal network (RPN) for multiple damage detection. Furthermore, Mundt et al. [114] developed a concrete defect bridge image dataset (CODEBRIM) of five commonly appearing concrete defects. They employed two meta-learning approaches based on reinforcement learning, i.e., MetaQNN and efficient neural architecture search, to find suitable CNN architectures for multi-class and multi-target damage detection.

The above ML and DL approaches are all based on inductive supervised learning, in which the performance relies on the pre-collected annotated examples before the inspection. They must work with pre-trained models to detect specific types of damage and cannot adapt themselves to novel defects quickly. However, annotation is usually time-consuming and tedious, and collecting sufficient defect images from various damage scenarios is not always practical. Traditional supervised transfer learning was expected to solve this issue, but it tends to be overfitting or challenging in convergence with only a few labelled examples. Little research exists about weakly supervised learning for few-shot image-based bridge damage classification and detection. The only related one is an attribute-based approach [115] for structural damage identification through meta-learning, which relies on episodic training through a series of pre-collected tasks and is not developed to the level of damage detection.

In summary, the previous research about damage detection and their approaches are illustrated in Table 1. As can be seen, proposing an efficient transductive FSL approach, which can be exempt from episodic training, is beneficial to assist vision-based bridge damage detection without a tedious data acquisition process before the inspection. It will also promise fast implementation for damage detection under complex circumstances with weakly supervised information.

Table 1 Related works for image-based structural damage detection

Names	Approaches	Research	Advantage / Disadvantage
Supervised Learning (Inductive)	Traditional ML	[14]–[26]	Fast with good interpretability but require handcrafted features
	DL	[13], [31]–[38]	No need for handcrafted features but heavy and require time-consuming image acquisition
Few-shot Learning (Weakly Supervised)	Meta-learning	[115]	Transductive inference with only a few examples but requires episodic training

2.3.1.2 Few-shot learning for images

The time-consuming and labour-intensive data acquisition process is the bottleneck for applying supervised ML in many fields. FSL aims to solve this issue by learning from a limited number of annotated images, including few-shot classification and segmentation, which is essentially related to the data-efficiency problem. This work focuses on the few-shot classification, which is usually taken as an example of meta-learning. A meta-learner is trained through a series of related works (episodic training) to perform well to unseen but related tasks with just a few examples. Meanwhile, transduction has been widely adopted for FSL tasks in learning and inference because it is more effective at using only a few labelled examples than induction with supervised models [116].

Many great efforts have been made in this field, including a few specific image datasets [117]–[120] (such as Omniglot, CIFAR-FS, CUB, and mini-ImageNet) and various approaches. For example, a few works [121]–[125] aim to use data augmentation based on different methods to solve the few-shot classification with limited training samples, such as self-augmentation [123], deformation [125], and generation from DCGAN [121]. Some other works aim to learn good model initialization [126], [127] or an optimizer [128], [129] to achieve rapid adaption with a limited number of training examples for new classes. In contrast, the other approaches aim to learn latent embeddings that can be used to compare [130] or cluster [131] query items using appropriate metrics. It includes creating the exemplar for each class from the support set and selecting a metric for evaluation [132]. For example, ProtoNet [133] calculates the mean vector of feature embedding as the prototype for each class in the support set and classifies query items as the nearest prototype based on the Euclidean distance because its case study fits Bregman divergence [134]. Some other approaches prefer cosine similarity [135], [136]. Relation Network further developed the ProtoNet using a relation module as a learning metric in training [137].

However, the sophisticated meta-learning FSL approaches are based on episodic training through an intentionally collected series of related works, which is still time-consuming. Recently, a few works [136], [138] have challenged the efficiency and effectiveness of this way by replacing episodic training with inter-class transfer learning (except the classes in the target FSL tasks). They can achieve similar state-of-the-art performance as the meta-learning approaches in the CUB and mini-ImageNet datasets. Furthermore, they have also indicated that the proper feature embeddings learned from cross-domain transfer learning (e.g., CUB → mini-ImageNet) can achieve competitive performance for FSL to the sophisticated meta-learning approaches [136], [138]. Moreover, the latest work [139] has demonstrated the availability of cross-domain transfer learning (i.e., ImageNet → MSCOCO and PASCAL VOC) for few-shot segmentation. It is achieved by leveraging a "training-tree" module (i.e., a pre-trained CNN backbone from ImageNet) to learn the feature representation.

Therefore, leveraging cross-domain transfer learning for few-shot damage detection is promising. However, the domain differences in the previous studies [136], [138],

[139] are not distinct enough compared to the domain difference from a public dataset to a specific engineering scenario, such as ImageNet → bridge structural defects (e.g., cracks, spalling, and corrosion). Therefore, this work aims to develop a transductive FSL approach for bridge damage detection using cross-domain transfer learning from a public dataset. It should be available for fast implementation under practical scenarios without episodic training and supervised learning, i.e., achieve similar "training-free" [139]. Hence, it is necessary to find a reliable source domain to perform effective feature embedding for few-shot damage detection and compare the performance of different pre-trained DNN backbones derived from supervised or unsupervised learning. Based on the transduction in the ProtoNet, it is also helpful to explore the performance of different metrics (i.e., Euclidean distance and cosine similarity) and propose a proper fine-tuning method for practical application.

2.3.2 3D point-cloud based survey

2.3.2.1 3D Damage detection and characterization

Although image-based damage detection and characterization have significantly progressed, they cannot provide the necessary depth information for 3D damage assessment and geometric model updating. The high-resolution point cloud, derived from photogrammetry, depth cameras, binocular cameras, terrestrial laser scanning (TLS), etc., is expected to fill this gap. For example, spalling and crack can be detected using the luminance or depth variation of RGB point-cloud data from TLS [140], [141] and then characterized approximately with the maximum depth. However, these approaches cannot distinguish between natural 3D damage and legitimate concave patterns, such as handcrafted circles, brick joints, and decorative textures. Hence, the latest research proposed a method based on mask R-CNN to achieve damage detection and segmentation. The mask geo-reference is achieved by the fusion of the image and point cloud [142]. This method can automatically distinguish target damage from the designed concaves. Still, it requires extra photo acquisition and complex coordinate transformation [143], which will bring in potential errors, especially when the image and the point cloud are obtained under different conditions (i.e., from different angles and distances). Therefore, developing an approach for surface damage detection and segmentation based on only the point cloud would be helpful.

2.3.2.2 Data processing and 3D damage assessment

Point-cloud data processing approaches can be categorized into point-based and voxel-based. Previous spatial damage detection and assessment studies mainly belong to the former, which compares the point cloud with the ideal 3D model or the previous captures. For example, the point cloud of a damaged reinforced concrete (RC) column after seismic testing can be sliced into different layers and compared with a rectangle to distinguish the spalling and residual areas [144]. Component point-cloud captures at different times can be compared with each other to identify damage and monitor its evolution [145]. Moreover, the damage volumetric quantification can be achieved with convex-hull fitting [142] based on point cloud. However, it is prone to exaggerate the ground truth of the target volume for non-convex geometry, so it requires elaborate manual separation for each component, as the dash lines shown in Figure 7, which is not always practical for real-world damage with complicated geometry. The point-based approaches lose sight of the benefit of voxels for volume calculation and mitigation of point sparsity. Therefore, developing a voxel-based point-cloud processing approach for 3D damage detection and assessment would be helpful.

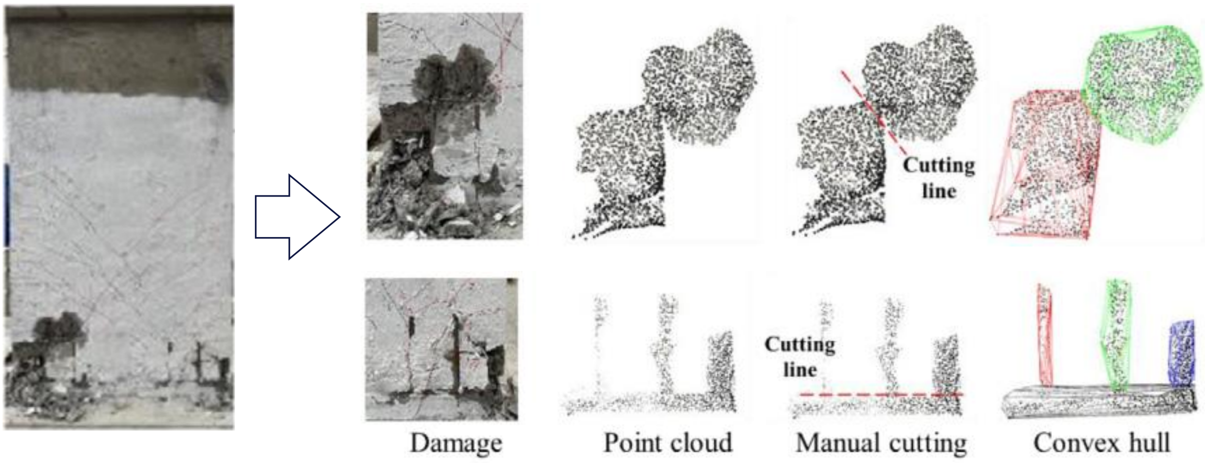


Figure 7 Convex-hull fitting for damage quantification based on manual cutting [142]

2.3.2.3 Model synchronization and data compression

The infrastructure DT for structure health monitoring (SHM) and maintenance is not only about 3D visualization but also involves communication and back-end services, such as FEA and BIM, as well as feedback to the physical entity (PE). From the practitioners' view, one obstacle to digital-twin application in practice is the difficulty of keeping model synchronization automatically in routine inspection [41]. The current 3D survey for local damage is still far from automated, and the DT models cannot be updated in time, so developing an approach for "as-is" model updating with the detected local damage during the survey would improve the work efficiency. One of the bottlenecks is the communication complexity, i.e., the massive volume of point cloud data. It is a heavy load for data transmission in both time and cost, especially for some infrastructures under circumstances with limited communication, such as bridges and tunnels. Therefore, it is necessary to only transmit the damaged part in an efficient format instead of the raw point cloud.

Additionally, compression can be used to further decrease communication complexity. For example, previous studies proposed a few approaches to compress 3D point clouds using RNN with residual blocks [146] or a hierarchical auto encoder [147]. However, the transmitted data for DT synchronization is not necessarily a point cloud. It can be any format that distinguishes the 3D damaged space from the residual entity. Therefore, binarized voxels have become a promising way to reduce complexity significantly, and lossless run-length encoding (RLE) can be utilized for further compression. In the medical field, the 3D binary matrix through the volumetric RLE has already been proven successful as an efficient approach for transmitting 3D medical imaging data, as shown in Figure 8.

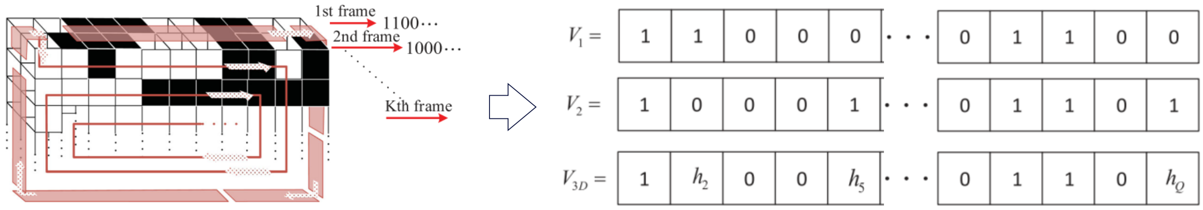

1st frame
1100···
2nd frame
1000···
Kth frame
$V_1 =$ 1 1 0 0 0 ··· 0 1 1 0 0
$V_2 =$ 1 0 0 0 1 ··· 0 1 1 0 1
$V_{3D} =$ 1 h_2 0 0 h_5 ··· 0 1 1 0 h_Q

Figure 8 Compression through 3D run-length encoding [148]

Moreover, the transmitted binarized voxels can update the FE and BIM models. The latest research [142] has demonstrated the effectiveness of updating the FE model by deleting the corresponding elements less than the damage depth. The method is based on iteration and can be integrated into mainstream FE software programs. Moreover, the previous studies [149]–[152] have revealed the availability and workflow for damage modelling in geometry and semantics using IFC files.

2.4 IFD and health monitoring with time series

2.4.1 Traditional machine learning pipeline

IFD methods that can automatically recognise the health states of machines and infrastructures [153] are essential for preventative maintenance in Industry 4.0. Many traditional ML approaches can be applied in IFD, such as k-nearest Neighbour (k-NN) [154], Naïve Bayes classifier [155], support vector machine (SVM) [156], decision tree [157] and random forests [158], etc., which rely on manual features. The pipeline for IFD based on traditional ML can be condensed as shown in Figure 9, which starts from data acquisition through various IoT technologies to feature extraction via handcrafted design and automatic data-driven health state recognition using supervised or unsupervised learning approaches.

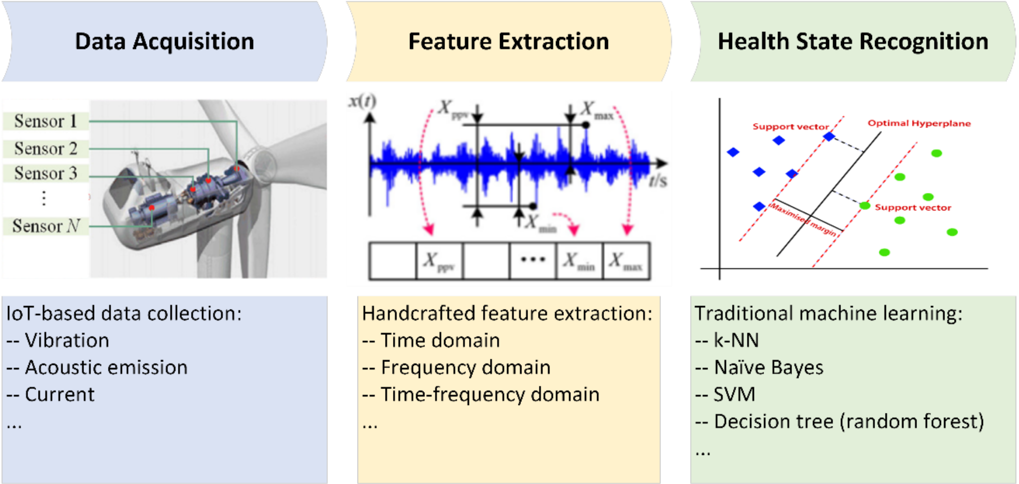


Figure 9 The IFD pipeline through traditional machine learning [153]

Data for fault diagnosis are usually in time series and collected constantly from different sensors mounted on machines or infrastructures, such as acceleration, displacement, strain, and acoustic signals, as well as ambient conditions like temperature and wind speed. The commonly used features can be categorised into time, frequency, and time-frequency domains based on the extraction methods, e.g., the statistical features, zero-cross rate, wavelet, fractal features in the time domain; discrete Fourier transform (DFT) and power spectral density (PSD) in the frequency domain; energy and entropy from short-term Fourier transform (STFT), wavelet transform (WT), wave packet transform (WPT), and Hilbert-Huang transform (HHT) in the time-frequency domain, as shown in Table 2.

Table 2 Traditional machine learning pipeline for IFD

Machine learning	Handcrafted Feature Extraction	Approaches
Traditional ML	Time domain: statistical features, zero-cross rate, wavelet, fractal features, etc.	KNN, SVM, Naïve Bayes Classifier, decision tree, random forest, etc.
	Frequency domain: DFT, PSD, etc.	
	Time-frequency domain: STFT, WT, WPT, EMD, HTT, etc.	

2.4.2 Deep learning pipeline

With the rapid development of IoT, the collected data volume is dramatically higher than ever before and brings more useful information for fault diagnosis. Big data acquisition has four characteristics: volume, quality, variety, and velocity [153].

(1) Volume – the volume of collected data sustainably grows during the long-term O&M.

(2) Quality – a portion of poor-quality data is mingled in the massive data.

(3) Variety – multi-source data is collected from multiple sources (by different sensors) with a heterogeneous structure.

(4) Velocity – fast transmission can be enabled in situ via fieldbus cables or at the remote end via high-speed communication like 5G, which promises response and decision-making in near real-time for DT.

Traditional ML relying on handcrafted features becomes inappropriate for big data scenarios. Hence, IFD has been extensively developed based on DL, which can learn features automatically. Its pipeline is shown in Figure 10, consisting of only two steps, i.e., data acquisition and health state recognition, which can accommodate massive data and achieve a higher level of automation by skipping the step of manual feature extraction. The widely used DL approaches for IFD include multilayer perceptron (MLP), autoencoder (AE), recurrent neural network (RNN), CNN, transformer, etc.

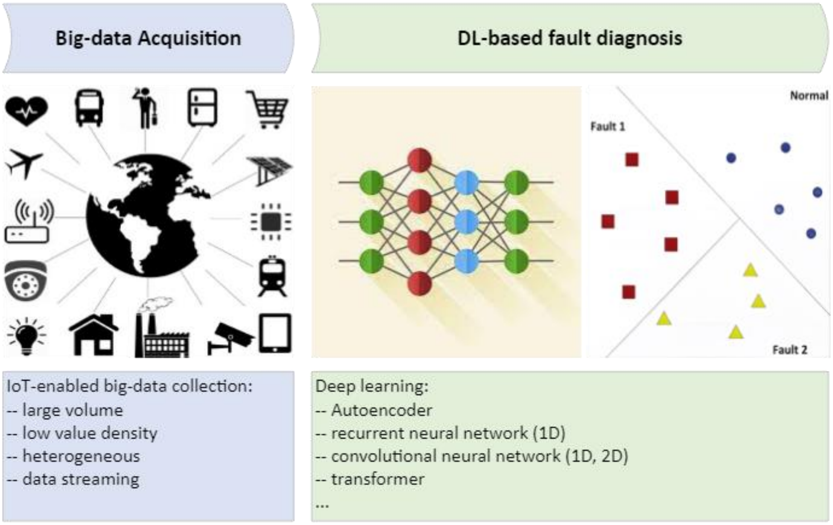


Figure 10 IFD pipeline through deep learning [153]

2.4.2.1 DL with 1D time series

Liu et al. [159] and Lu et al. [160] employed the stacked sparse AE and the stacked denoising AE for the IFD of bearings, presenting higher diagnosis accuracy than traditional ML methods. Common RNNs, including gated recurrent units (GRUs) and long-term memory networks (LSTM), are theoretically an ideal non-linear time-series forecasting tool and a universal approximator for dynamic systems [161]. Ling et al. [162] employed RNN to achieve early warning in the fault creep period for nuclear power machinery, together with principal component analysis (PCA), wavelet analysis, and Bayesian inference. Yuan et al. [163] utilised LSTM for IFD and remaining useful life (RUL) estimation for aero-engine based on time-series data. Moreover, Neves et al. [164], [165] employed an MLP to identify the structure health conditions of the KW51 railway bridge with train-induced acceleration data. Sajedi and Liang [166] proposed a framework based on a fully convolutional encoder-decoder architecture for structural damage diagnosis with the vibration signals from a grid sensor network, which can localise damages and distinguish multiple damage mechanisms with reliable generalisation capacities.

Additionally, 1D-CNN is also inherently suitable for time-series pattern recognition. For example, Wu et al. [167] proposed an approach for rub-impact fault diagnosis of a rotor system based on 1D-CNN. Sony et al. [168] designed a 1D-CNN to identify multiclass damage using bridge vibration data. 1D CNN was also utilised to detect the change of local structural stiffness and mass based on acceleration from a single sensor [169], [170].

2.4.2.2 DL with 2D synthetic images

As the monitoring variable for IFD is usually a 1D time series, which is different from 2D images, to leverage the powerful feature learning capability of CNNs, many efforts have been made to transform 1D motion signals into 2D images, including Gramian angular field (GAF) [171], wavelet transform [172]–[174], S-transform [175], phase space reconstruction [176], etc. The GAF, wavelet transform, and S-transform are time-consuming, and the latter two require expert knowledge in the frequency domain for spectrum exploration. In contrast, phase space reconstruction can quickly generate synthetic images with simple backgrounds. For example, time series can be converted through Eq. 1. (i.e., min-max normalisation) into a single-channel greyscale image, as shown in Figure 11.

$$P(j,k) = round\left\{\frac{f\left(x\left((j-1)\times L+k\right)-Min(f(x)\right)}{Max(f(x))-Min(f(x))}\times 255\right\} \tag{1}$$

Where $P(j,k) \in [0,255]$ denotes the pixel strength of the grayscale image; j and k are the row and column numbers in the reconstructed image, respectively.

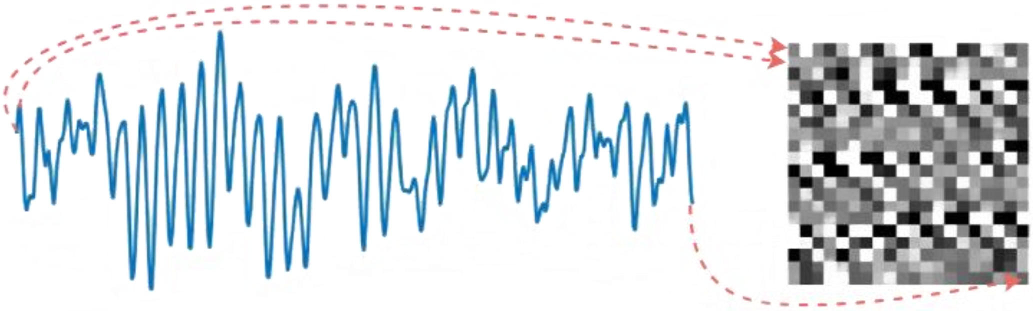


Figure 11 Reconstruction from time series to a single-channel grayscale image [176]

The DL-based IFD can be summarised as shown in Table 3. Previous works [176]–[178] have already proved the effectiveness of using shallow CNNs, like modified LeNet, for IFD. However, they mainly focused on a single sensor and did not consider data fusion for the signals from triple sensors or axes. Meanwhile, the imaging method has not been further developed to generate 3-channel images (like RGB) to take advantage of the popular deep CNN architectures.

Table 3 Deep learning pipeline for IFD

Pipeline	Approaches
Deep Learning	1D time series: RNN (including GRU and LSTM), 1D-CNN, etc. 2D synthetic images: 1) Imaging – GAF, wavelet transform, S-transform, phase space reconstruction, etc. 2) Models – shallow single-channel CNNs and classical 3-channel deep CNNs via proposed imaging.

2.4.3 Multi-sensor data fusion

Data fusion is usually employed in IFD based on multi-sensor data, which is supposed to be an effective way to improve pattern recognition accuracy. It includes data-level and decision-level fusion. Teng et al. [179] trained seven individual 1D CNNs using the acceleration signals from the corresponding sensors and fused their classification results at the decision level by hard voting. Compared with data-level fusion, i.e., integrating all acceleration signals into a multi-channel time sequence, decision-level fusion enhanced the classification accuracy by at least 10% in the experiments. However, this comparison consequence is not absolute. For example, Gao et al. [9] trained a single 1D CNN with the data-level fused acceleration signals from six sensors on a bridge for structure health-state recognition. Compared with decision-level fusion with hard and soft voting from six individual classifiers, data-level fusion can enhance test accuracy by more than 20%. Furthermore, Gong et al. [180] used a multi-channel data-level fusion of time series signals from different sensors for the IFD of rotating machinery by leveraging CNN-SVM, which also achieves excellent test performance (nearly 100% accuracy). As can be seen, the level of fusion occurrence in IFD is flexible, depending on the used dataset and the selected neural network architecture.

2.5 Bridge maintenance knowledge graph

2.5.1.1 Bridge knowledge engineering

KE is a field of AI that creates rules by applying data to imitate the thought process of a human expert. It looks at the structure of a task or a decision to identify how a conclusion is reached [181]. In the AEC industry, KE refers to knowledge representation, acquisition, reasoning, decision-making, and application in building, operation, and maintenance. A knowledge graph (KG) has become one of the most effective tools for knowledge management and integration. For most studies in the construction fields, KGs are usually termed ontological semantic networks based on graphics with domain entities as nodes and the defined entities' relationships as edges [182].

To represent the intricate knowledge of bridge maintenance, Ren et al. [183]

developed an ontology for bridge maintenance (BrMontology) based on web ontology language (OWL), which covers bridge structure, damage (and causes), solutions, and big events. It can enable automatic rule-based maintenance planning and holistic decision-making. Liu and ElGohary [184] proposed a bridge ontology (BridgeOnto) based on routine inspection reports and maintenance manuals in the US, in which the bridge elements, types of bridge elements, and bridge defects are elaborately decomposed. Zhang et al. [182] proposed a comprehensive BMKG involving expenses, which utilized an ontology for knowledge organization and the graph database Neo4j for storage. Wu et al. [185] developed an ontology for project management of bridge rehabilitation, which covers restoration tasks and constraints.

Additionally, several studies focused on a knowledge-based bridge O&M system, especially by leveraging data mining. For example, Li et al. [186] designed a bridge structure health monitoring (SHM) system based on a fine-grained ontology to integrate heterogeneous data from various sensors. Yang et al. [187] developed a framework for bridge management based on a big-data KE paradigm, which consists of the layers of data (sources, storage, and computing) and knowledge (representation, computing, and services). This framework can facilitate intelligent bridge maintenance by leveraging the big data within the entire bridge life cycle.

Moreover, some efforts have been made for automatic ontology generation and KG completion, such as semantic information extraction from inspection reports, including named entity recognition for target classes [188], [189] (e.g., bridge element, deficiency, cause, repair action) and dependency parsing [190], [191] (i.e., relation extraction) from textual reports. These works utilized NLP approaches to build (or complete) an ontology. Still, the data mining of the generated KG usually relies on classic algorithms through the graph properties, such as centrality, clustering, and pathfinding. However, few studies explored using node or edge text embedding and graph neural network (GNN) in the BMKG for semantic enrichment and maintenance recommendation.

2.5.1.2 Graph data mining with GNN

Although the classic DNNs achieve great success for latent embedding from Euclidean spatial data, they cannot perform satisfactorily in processing non-Euclidean data, such as graphs. Hence, GNNs are proposed to solve this issue (i.e., graph embedding), which are defined as an optimizable transformation on all attributes of the graph (nodes, edges, global context) that preserves graph symmetries (permutation invariances) [192]. Graph embedding is a process to generate a vector from graph features and attributes but tries to preserve graph information as far as possible so that the downstream graph analytic tasks can be achieved easily using the off-the-self ML algorithms [193]. Embedding aggregation and message passing are the primary attributes of GNNs. The common GNN architectures [194] include graph convolutional networks (GCN), graph attention networks (GAT), GraphSAGE, graph spatial-temporal networks, etc. Most GNNs are based on transductive learning, in which the model is trained using the whole graph. In contrast, others (such as GraphSAGE) rely on inductive learning, where the model can only see the training data. Thus, the generated model can predict graph labels for unseen data.

GNNs are a set of neural networks that operate on graph data, and they can usually be applied to graph classification, node classification, link prediction, etc. Collins et al. [195] applied graph classification through GCN to classify IFC building objects based on their geometry in the AEC industry. Wang et al. [196] proposed an approach for the semantic enrichment of BIM models by using node classification with GraphSAGE for room types. Moreover, link prediction is usually utilized for recommendation and KG completion. Xia et al. [197] developed an approach to recommend potential solutions and explain the fault for oil drilling equipment using linkage prediction through an attention-based compressed relational GCN (ACRGCN), inspiring this work.

GNNs have demonstrated their potential in context-related NLP tasks, such as text similarity calculation and sentence completion. However, few efforts have been made on the BMKG about graph data mining. Hence, it would be helpful to 1) propose a maintenance-oriented KG schema based on real-world bridge maintenance reports and 2) explore the GNN algorithms with text embeddings as node features on the generated BMKG for graph data mining, such as contextual node classification and link prediction for both semantic enrichment and maintenance recommendation.

2.6 Summary of literature findings

This chapter introduced the research status quo in the field of DT for bridges in terms of different purposes, such as bridge SHM, O&M, and asset management, etc., including conceptual frameworks and implemented systems, and bridge-machine-human triplet interaction, as well as the representative technologies related to bridge maintenance, which aims to be integrated into the target bridge DT of this thesis. The content highlights the patterns and contributions of the existing studies for bridge DT and identifies the gaps between academic research and practical implementation, inspiring this research. Several findings can be illustrated below:

Firstly, In the context of industry 4.0, the DT has received extensive attention for improving the AEC processes and the product life-cycle management, such as bridge O&M. The ongoing research indicates a notable characteristic (or pattern) in the current studies that DT systems are always designed and developed for specific purposes and circumstances [17], [18], which is the principal hypothesis for any DT framework and system development.

Secondly, bridge DT-related technologies can be embodied in the triplet interaction between bridge, machine, and human. This novel triplet interaction can reflect technical targets, data streaming, and the locations of technology deployment (such as on the site of physical bridges, middleware, or use end), as well as provides a new perspective for DT understanding different from the previous one between PE and VE, which can further benefit the DT conceptual design and practical implementation.

Thirdly, although DT technologies have made advances in many areas, such as aerospace and smart manufacturing, these successful precedents (including frameworks and systems) cannot be transferred to bridges directly because when a DT is implemented on the bridges, their locations may concern with specific issues, e.g., the remote bridges might not have the required access to communication or electricity for massive data transmission. Therefore, it is desired that a bridge DT would have an efficient and resilient framework that can work under communication-constraint circumstances and endure the loss of signal or similar communication breakdown problems. Moreover, DT integration of general bridge maintenance workflow, data streaming among different platforms (or middleware), and a working prototype are also significant challenges.

Fourthly, most bridge DT studies focus on a single aspect of the triplet interaction, such as bridge-machine interaction (like drone-enable bridge inspection and IoT-based SHM) and human-bridge interaction (like optimization-oriented bridge maintenance). The former provides data to create a digital representation with early warning and simple assessment, while the latter leverages data, information, and knowledge from multiple sources for holistic decision-making. However, there is a scarcity of research that explains mutual connections clearly and seamlessly incorporates them into a unified bridge DT system. Therefore, the practical survey and SHM in most bridge DTs are still discrete (project-based or condition-based), which cannot take full advantage of the DT concept by triggering various backend DT services and providing timely feedback to the side of physical bridges.

Fifthly, there is no study for bridge DT setting its scope for the complete bridge maintenance cycle using a general workflow, i.e., from on-site inspection and monitoring to bridge maintenance planning and decision-making. In this scope, many practical issues can be raised related to efficiency:

1) The current automatic visual inspection is mainly based on supervised learning,

requiring time-consuming and labour-intensive image acquisition. However, it is not always practical, especially under complex circumstances.

2) The TLS survey is still project-based due to the heavy load of point-cloud data and cannot achieve timely model synchronization and damage assessment for DT backend services.

3) Although many individual approaches have been developed for IFD and SHM based on multi-sensor time series signals, there is still a lack of a unified framework leveraging AutoML and data fusion for network search and parameter optimization to achieve autonomous DL-based health-state recognition for the long term.

4) There are few KG schemas designed based on practical bridge inspection reports for bridge maintenance, and the practical bridge maintenance suggestion is still a time-consuming workflow based on experienced engineers. In contrast, the graph-based homogeneous bridge structures (such as symmetry and repeatability) and the text similarity based on domain knowledge in the historical reports are neglected, which can be leveraged to improve the working efficiency via the analogical strategy.

Chapter 3 Research Methodology

This chapter presents the overarching methodology through the research, which can be condensed into the “Research Opinion” framework [198]. The procedure is explained in the following sections, from the outer layers to the inner layers, including philosophies, approaches, strategies, choices, time horizons, techniques and procedures, as shown in Figure 12. Then, the research roadmap is presented at the end, which echoes the above methodology framework and components.

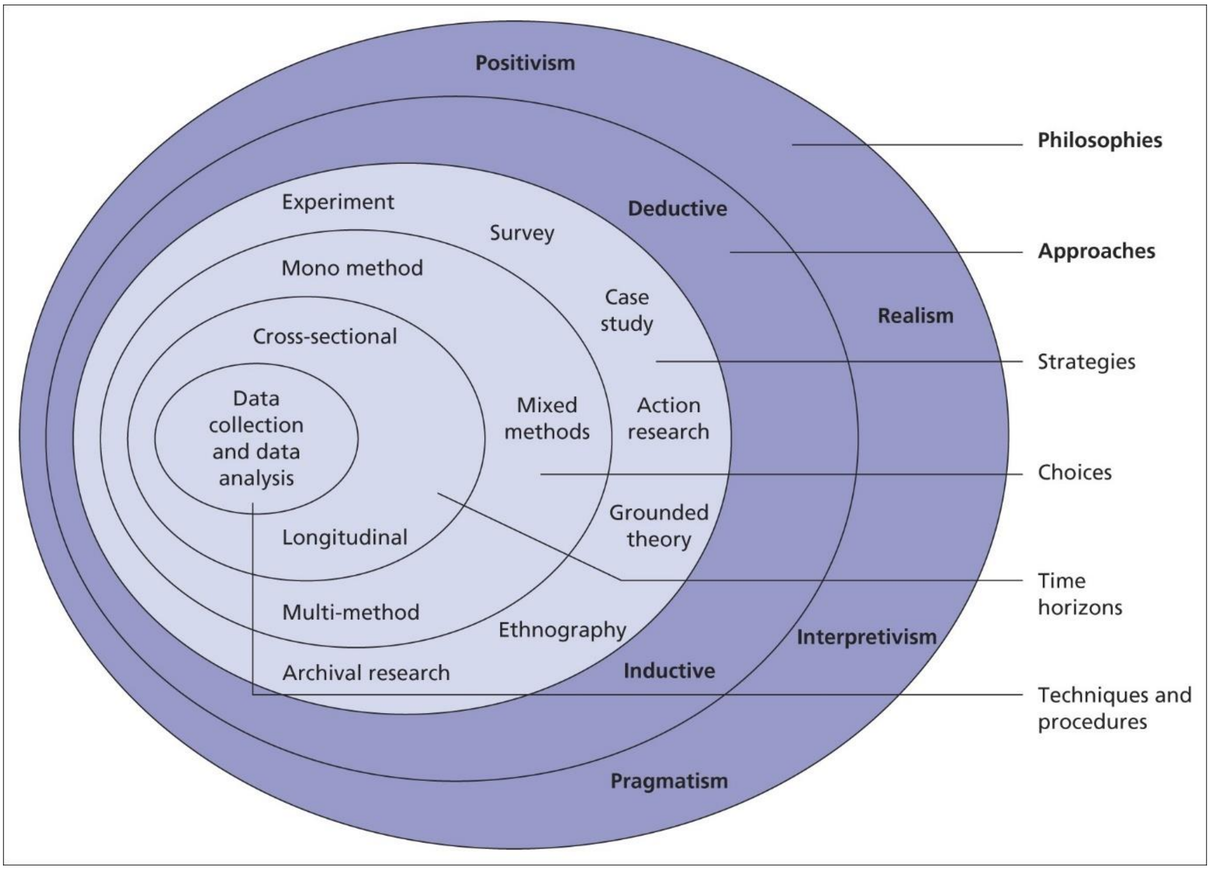


Figure 12 The “research onion” [198]

3.1 Research philosophies

Philosophy refers to the collection of principles that dictate the perspective or standpoint from which research is undertaken. It is usually studied in terms of metaphysics and epistemology [199]. The former refers to the authenticity of the information and how one understands its existence. In contrast, the latter refers to the valid information required for the research and how to obtain it [200], the core philosophy theory of knowledge in this research.

Philosophical positions used in academic studies are often divided into positivism, interpretivism, critical realism, and pragmatism [198]. Positivism assumes that knowledge is independent of the subject being studied, which is often more scientific in testing phenomena; interpretivism claims that individual observers have their perception and understanding of reality; critical realism emphasizes historical causal explanation as construction; pragmatism focuses on practical problem-solving and future-oriented practice as construction [200], [201]. Hence, positivism and pragmatism are employed in this research as the major research philosophy paradigms, whilst there are also a few elements of the others for specific research questions.

3.2 Research approaches, strategies and choices

The research approaches can be categorized into deductive and inductive [198]. The deductive approach starts with a specific hypothesis development based on the literature review that the researcher has observed. It tries to gradually test this hypothesis and checks if the hypothesis holds in particular contexts [200]. In contrast, the inductive approach starts with observations that the researcher uses to create a new theory [200]. In this research, the former deductive one is chosen as the approach to commence the research for each question with appropriate hypotheses and validate these hypotheses and the discovered new knowledge based on them under particular circumstances.

Moreover, as indicated by Mackenzie and Knipe [202], there is no necessity to define a single specific strategy to achieve research objectives. By contrast, the multiple strategies in most cases can enable a well-structured research design to a greater extent. Hence, various strategies, including systematic literature review, survey, case study, and grounded theory, are utilized across different research questions. Moreover, both quantitative and qualitative methods can significantly contribute to this pragmatic research. Therefore, the mixed methods are adopted in this research.

3.3 Research time horizons, techniques and procedures

Research time horizon refers to the time frame of the research [198]. Generally, observations can be of two types based on time horizons: cross-sectional and longitudinal. The cross-sectional data is used when all observations are for a single point of time, such as in most surveys [200]. Longitudinal data, in contrast, implies the observations for a particular variable that is available for several years, quarters, months or days [200]. This research mainly focuses on the former, i.e., cross-sectional studies. At the same time, the latter is expected in future work to test the outcomes from this research (such as frameworks, approaches, and new knowledge) along the time dimension.

Finally, the last layer of the “research onion” consists of the techniques and procedures used [198], which aim to clearly explain the ways and purposes of the conducted research. As data is considered the central piece in the “research onion” model, it is expected to choose between primary and secondary data or qualitative and quantitative data collected from different sources at this stage [201]. This research employs both primary data (e.g., interviews, questionnaires, and measurements) and secondary data (e.g., literature, statistics, public datasets). Moreover, as the mixed methods are selected for this pragmatical research, the data and information are derived from both quantitative and qualitative sources.

3.4 Overarching research roadmap

The overarching research roadmap can be condensed into Figure 13, in which the methodology details for each research question (i.e., each chapter) are illustrated along with hypotheses (i.e., H1, H2, … H5) and outcomes.

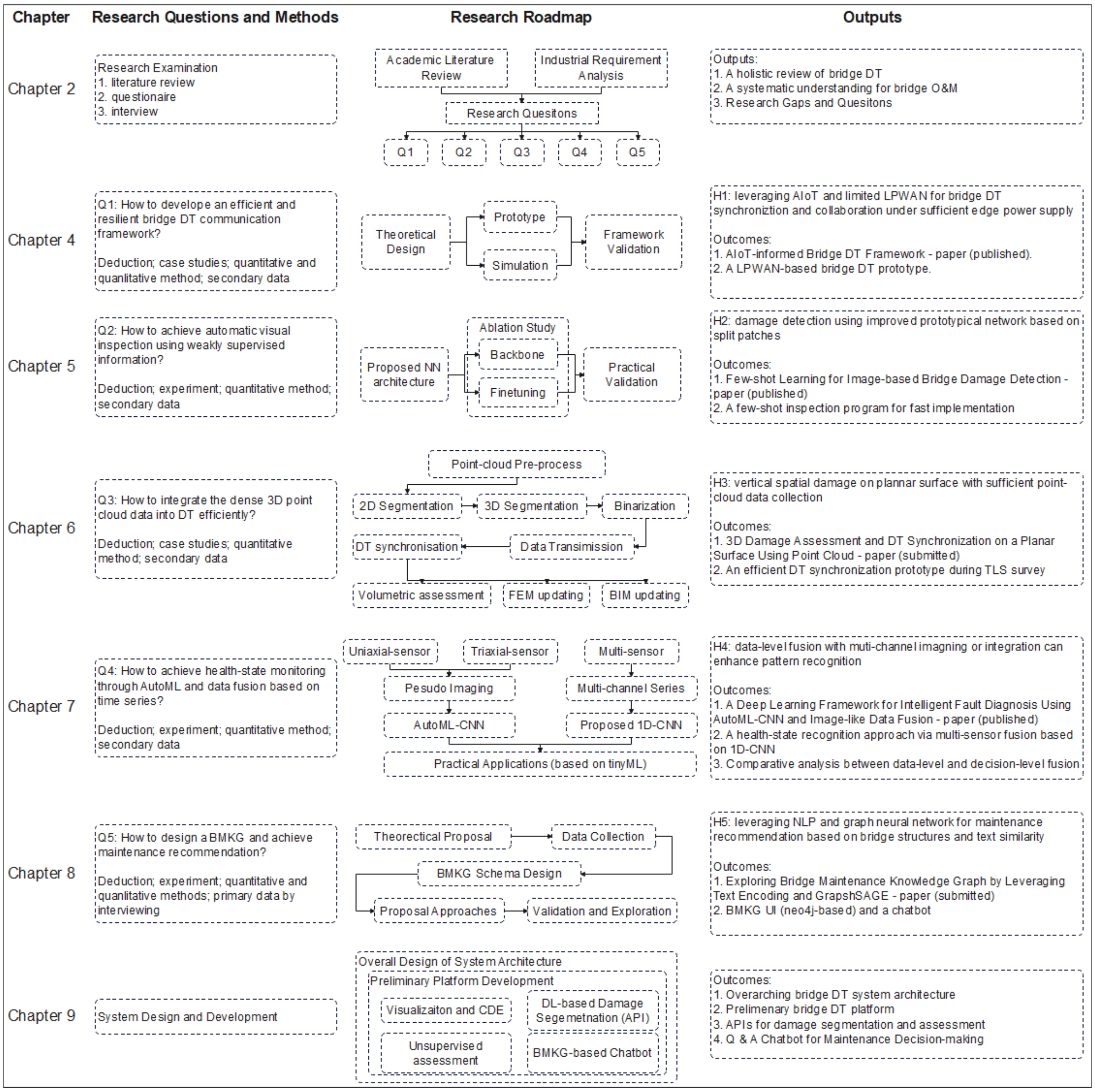

Chapter
Research Questions and Methods
Research Roadmap
Outputs
Chapter 2
Research Examination
1. literature review
2. questionaire
3. interview
Academic Literature Review
Industrial Requirement Analysis
Research Quesitons
Q1
Q2
Q3
Q4
Q5
Outputs:
1. A holistic review of bridge DT
2. A systematic understanding for bridge O&M
3. Research Gaps and Quesitons
Chapter 4
Q1: How to develope an efficient and resilient bridge DT communication framework?
Deduction; case studies; quantitative and quanlitative method; secondary data
Theoretical Design
Prototype
Simulation
Framework Validation
H1: leveraging AIoT and limited LPWAN for bridge DT synchroniztion and collaboration under sufficient edge power supply
Outcomes:
1. AIoT-informed Bridge DT Framework - paper (published).
2. A LPWAN-based bridge DT prototype.
Chapter 5
Q2: How to achieve automatic visual inspection using weakly supervised information?
Deduction; experiment; quantitative method; secondary data
Proposed NN architecture
Ablation Study
Backbone
Finetuning
Practical Validation
H2: damage detection using improved prototypical network based on split patches
Outcomes:
1. Few-shot Learning for Image-based Bridge Damage Detection - paper (published)
2. A few-shot inspection program for fast implementation
Chapter 6
Q3: How to integrate the dense 3D point cloud data into DT efficiently?
Deduction; case studies; quantitative method; secondary data
Point-cloud Pre-process
2D Segmentation
3D Segmentation
Binarization
DT synchronisation
Data Transimission
Volumetric assessment
FEM updating
BIM updating
H3: vertical spatial damage on plannar surface with sufficient point-cloud data collection
Outcomes:
1. 3D Damage Assessment and DT Synchronization on a Planar Surface Using Point Cloud - paper (submitted)
2. An efficient DT synchronization prototype during TLS survey
Chapter 7
Q4: How to achieve health-state monitoring through AutoML and data fusion based on time series?
Deduction; experiment; quantitative method; secondary data
Uniaxial-sensor
Triaxial-sensor
Multi-sensor
Pesudo Imaging
Multi-channel Series
AutoML-CNN
Proposed 1D-CNN
Practical Applications (based on tinyML)
H4: data-level fusion with muti-channel imagning or integration can enhance pattern recognition
Outcomes:
1. A Deep Learning Framework for Intelligent Fault Diagnosis Using AutoML-CNN and Image-like Data Fusion - paper (published)
2. A health-state recognition approach via multi-sensor fusion based on 1D-CNN
3. Comparative analysis between data-level and decision-level fusion
Chapter 8
Q5: How to design a BMKG and achieve maintenance recommendation?
Deduction; experiment; quantitative and quanlitative methods; primary data by interviewing
Theorectical Proposal
Data Collection
BMKG Schema Design
Proposal Approaches
Validation and Exploration
H5: leveraging NLP and graph neural network for maintenance recommendation based on bridge structures and text similarity
Outcomes:
1. Exploring Bridge Maintenance Knowledge Graph by Leveraging Text Encoding and GrapshSAGE - paper (submitted)
2. BMKG UI (neo4j-based) and a chatbot
Chapter 9
System Design and Development
Overall Design of System Architecture
Preliminary Platform Development
Visualizaiton and CDE
DL-based Damage Segemetnation (API)
Unsupervised assessment
BMKG-based Chatbot
Outcomes:
1. Overarching bridge DT system architecture
2. Prelimenary bridge DT platform
3. APIs for damage segmentation and assessment
4. Q & A Chatbot for Maintenance Decision-making

Figure 13 Overarching research roadmap

Chapter 4 AIoT-informed DT framework

DT has been moving progressively from concept to practice for bridge O&M, but its data synchronization and fault tolerance issues remain problematic. This chapter investigates the time delay of bridge DT services according to communication and computation complexity, revealing the distinct impact of their sequence, and proposes an AIoT-informed DT communication framework to solve the above issues. The information hierarchy and two-way communication can be leveraged to minimize communication complexity in the framework. Meanwhile, the data flow and resilience of the proposed framework are demonstrated using a Petri net. Moreover, the framework is developed into a prototypical DT through cross-platform integration and validated with different cases. The results indicate that compared with other existing bridge DTs, the proposed framework has high efficiency, low latency, and excellent fault tolerance, which can contribute to the efficiency and safety of bridge O&M, especially under communication-constraint circumstances. The framework is also promising for federated learning to protect the AI-model privacy of different stakeholders and can potentially support agent-based intelligent bridge management in the future with little human intervention.

4.1 Theoretical foundation

As seen from Section 2.2.2, massive heterogeneous data from regular inspection and real-time monitoring of physical bridges has become a challenge for bridge DT synchronization, especially under communication-constraint environments and for large-scale applications. Meanwhile, DT computation and performance are also influenced by the complexity of enormous multi-source data and inevitable system faults (e.g., loss of communication) in terms of time delay and resilience. Although some research [203], [204] explored edge computing and federated learning for SHM, which has relatively low communication complexity, they were not developed into a complete and comprehensive bridge DT system. Recently, the concept of Artificial

Intelligence of Things (AIoT) has received widespread attention, which combines AI technologies with the Internet of Things (IoT) infrastructures to achieve more efficient IoT operations, improve human-machine interactions and enhance data management and analytics [205]. This research aims to study the time delay, complexity, and fault tolerance of bridge DT and reveal how to design a bridge DT to overcome the barriers of data synchronization and communication faults theoretically, as well as develop an AIoT-informed DT communication framework to support bridge O&M with high efficiency, low latency, and excellent resiliency.

4.2 Time delay and complexity

Although there is always a time delay between PE and VE in a DT system, how much of it can be tolerable depends on pre-designed purposes (services) and practical application scenarios. For bridge operation, the low latency of DT services can enable timely "what-if" analysis and quick emergency response. Specifically, the time delay of DT services, such as early warnings and protective measures, is critical for the safety of physical bridges and the public travelling on the bridges when a disaster is happening or predicted to occur. Meanwhile, a short delay between the physical and virtual bridges can also enhance maintenance efficiency, such as inquiry, inference, and decision-making during the inspection, especially when a complex issue requires big-data analysis or involves multiple stakeholders.

Time delay of DT services comprises communication time T_{comm} and computation time T_{comp}, as indicated in Eq. 2.

$$T_{delay} = T_{comm} + T_{comp} \tag{2}$$

Where comm and comp represent communication and computation, respectively.

Computation time is directly proportional to computational complexity $O(n)$, as indicated in Eq. 3 [206], where n is the number of variables. Computational complexity includes time complexity, i.e., the time taken by the algorithm to execute each set of instructions, and space complexity, i.e., the amount of memory consumed by the algorithm [207].

$$T_{comp} \propto \frac{Computational\ Complexity}{Clock\ Frequency} \tag{3}$$

Where I/O and bus time for connecting peripheral devices are negligible.

In contrast, the computational time is inversely proportional to the clock frequency of processors when all the processes are sequential, i.e., only one pulse at one time on a single core. For example, suppose an algorithm running on a Y MHz processor takes t seconds to execute, then moves the same algorithm to a Z MHz processor. In that case, the program is expected to be completed in approximately $(Y/Z) \times t$ seconds [208]. Although it is no longer the case currently due to non-sequential ways (such as multi-core and multi-threading), this relationship is still valid.

Communication time is determined by communication complexity, bandwidth, and latency, as indicated in Eq. 4.

$$T_{comm} = \frac{Communcation\ Complexity}{Bandwidth} + Latency \tag{4}$$

Communication complexity (one-way or multiparty) is the amount of exchanged information (e.g., bits) among PE and VE necessary to perform the computation of certain DT services. The bandwidth is the maximum data transmission speed of a specific communication technology. Finally, the latency depends on the distance between communication nodes. Hence, the complexity determines the lower bound of the communication time, especially when communication is restricted with limited bandwidth, such as LPWAN. Therefore, reducing communication complexity becomes critical for decreasing time delay.

As data ambiguity and complexity are in direct proportion, i.e., more ambiguity means more complexity, the information hierarchy, i.e., the DIKW pyramid, can be leveraged to decrease complexity [209], as shown in Figure 14.

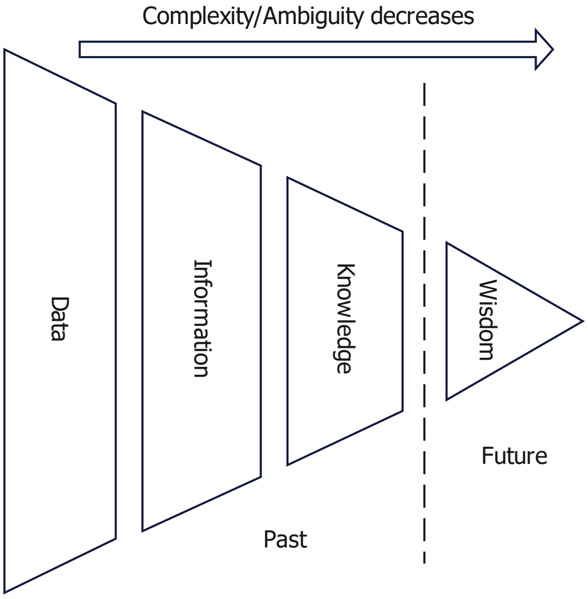


Figure 14 Complexity/ambiguity decreasing along the information hierarchy

Bridge routine inspection is a typical DIKW process, which can reduce ambiguity by extracting information and knowledge from inspection data based on the engineer's experience. Similarly, structural damage detection involves complexity reduction by obtaining information or knowledge from real-time SHM based on statistical models, big-data analysis, and ML. Moreover, knowledge from human engineers can be transferred to machines by AI (such as supervised or unsupervised learning) and KE, thereby achieving the automatic process to reduce complexity in the workflow. For example, agent-based drone inspection can contextualize bridge deficiency and ambient conditions automatically via localization, object detection, and semantic segmentation instead of human engineers [210].

Edge devices and the DT platform can be taken as two parties with different data sources, e.g., the former includes data collected on the site of physical bridges $(x \in X)$. At the same time, the latter involves multi-source data $(y \in Y)$ at the cloud server, such as historical records, inventory, weather, and natural disasters. Hence, it brings in an issue of how many bits they need to communicate with each other for computing the function f on $X \times Y$ until one party knows the value of $f(x,y)$ for decision-making. Here, cost(P) is the worst case of bits exchanged (maximum) for a protocol P, which can solve this problem over all inputs $(x,y) \in X \times Y$. Finally, the communication complexity $D(f)$ is determined with Eq. 5 [211].

$$D(f) = min\{\text{cost}(P) | P(x,y) = f(x,y) \text{ for all } (x,y) \in X \times Y\} \tag{5}$$

Note: x and y are n-bit strings; assume no concerns of computational power.

Theoretically, a communication protocol P can be defined as a rooted binary tree with internal nodes labelled by either E (edge) or C (cloud), as shown in Figure 15, indicating PE and VE, respectively. Each leaf has an output weight w in $\{0,1\}$ (bit exchanging or not). For simplicity, the function $f: X \times Y \rightarrow \{0,1\}^n$, encoded as a finite sequence of zeros and ones. f_v is associated with node v (if v is labelled by E, $f_v(x) \rightarrow \{0,1\}^m$, and vice versa). The bits at node a are sent by E with the calculated value $f_a(x)$, which is a binary string. The number of bits transmitted to C is $w_a \cdot (log_2(f(x)) + 1)$. Therefore, the number of bits exchanged to compute $f(x,y)$ in the path of $a \rightarrow b \rightarrow c \rightarrow d$ is $log_2(f_a(x)) + log_2(f_b(y)) + 1$.

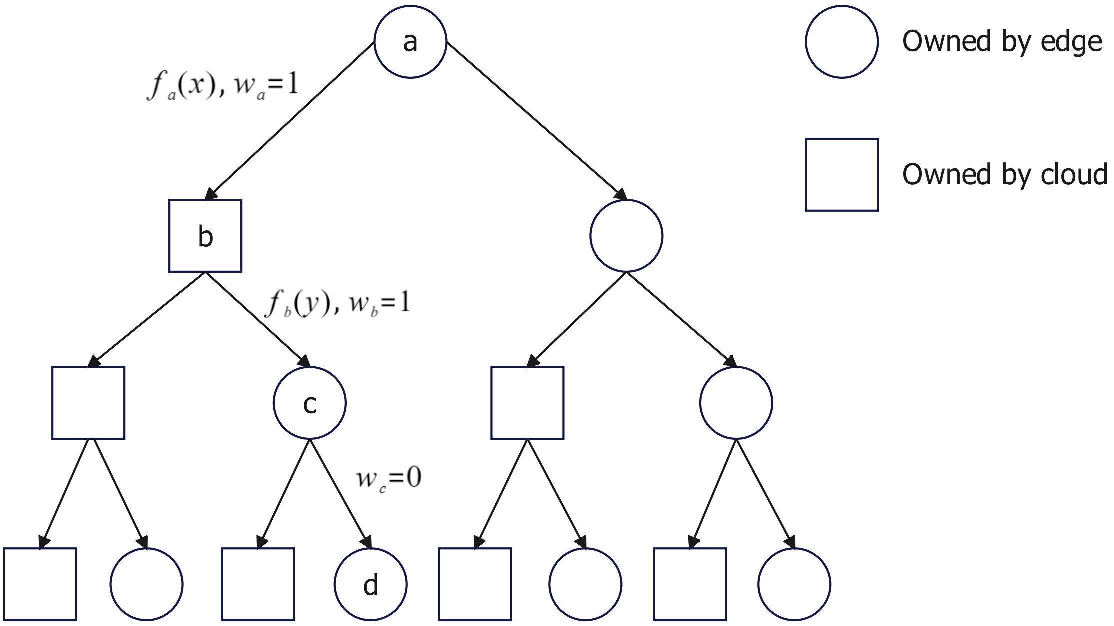


Figure 15 A rooted binary tree for computing $f(x,y)$ through two-way communication

There is always a simple protocol that sends all $x \in X$ to Y, performing the same as one-way communication, in which $D(f) = O(n)$. This way is the best one can do for the equality function (EQ), which outputs one if $x = y$. However, for other tasks requiring bidirectional communication, i.e., f computation is achieved by both parties, communication complexity can be decreased by a few methods, such as edge computing, so there is $D(f) \leq O(n)$. For example, for a parity function $\oplus_{2n}(x, y)$, the best way is to send $b = \oplus_n(x)$ to Y, then calculate $b \oplus (\oplus_n(y))$, in which $D(f)$ is only 1 bit.

In practice, many tasks do not always need to consume the maximum bits exchanged under the worst case. For example, for the structural assessment process (query problem) based on a decision tree, as shown in Figure 16, its cost(P) follows the longest path of the tree. However, many non-severe defect assessments do not need as much complexity as cost(P), so we can use hierarchical or interactive data exchange to reduce the practical communication. Furthermore, communication complexity can also be reduced by turning deterministic communication complexity $D(f)$ into randomized communication complexity $Pr[R(x)]$ even in one-way communication, e.g., only update the changing data beyond a threshold θ to the DT for synchronization as long as θ can meet the precision requirement of services.

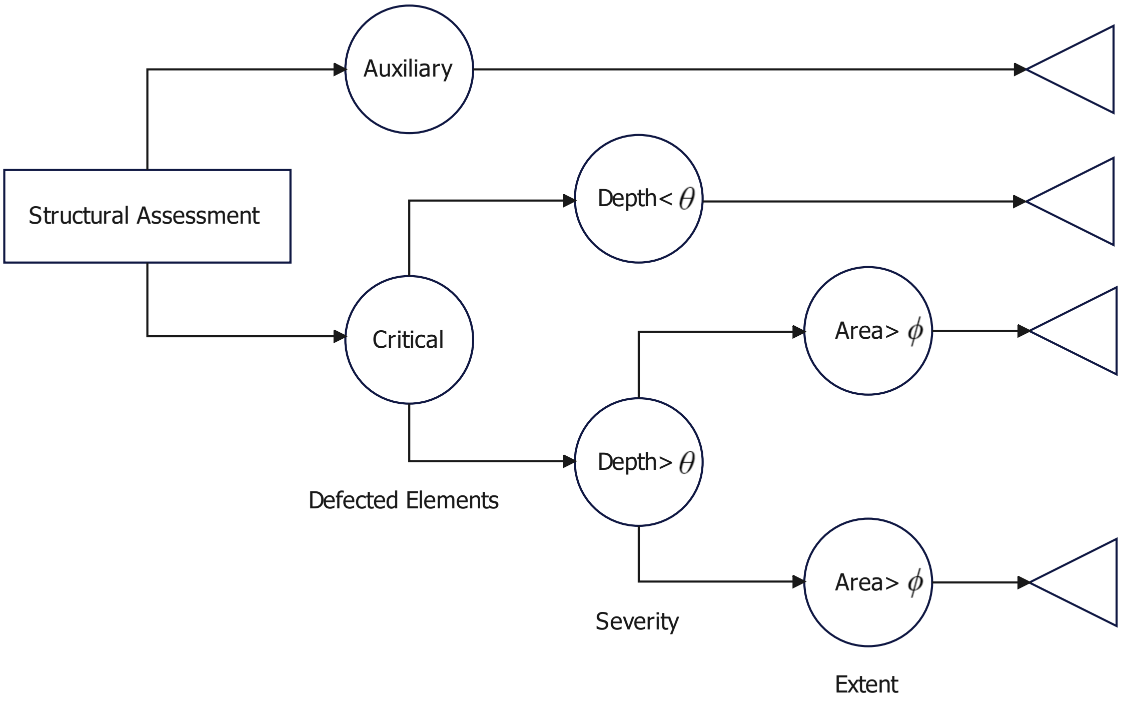


Figure 16 Bridge structural assessment based on a decision tree

Moreover, as T_{Delay} is comprised of $T_{communication}$ and $T_{computatiom}$, the sequence of communication and computing can also have a distinct impact on T_{Delay} of DT services, even if communication and computation complexity is constant, i.e., with a certain protocol and algorithm, as shown in Figure 17. Such impact can be amplified when either of $T_{communication}$ and $T_{computation}$ plays a dominant role in the process, e.g., when communication protocols are pretty restricted, such as LPWAN. For such a case, the total time delay T_{Delay} can be reduced significantly with appropriate edge computing before communication.

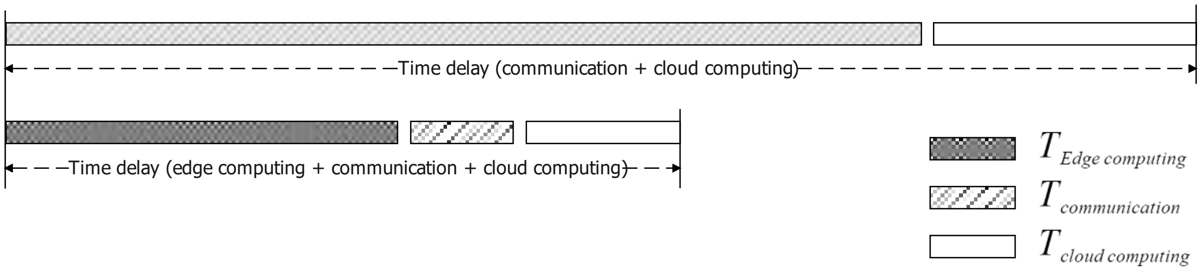
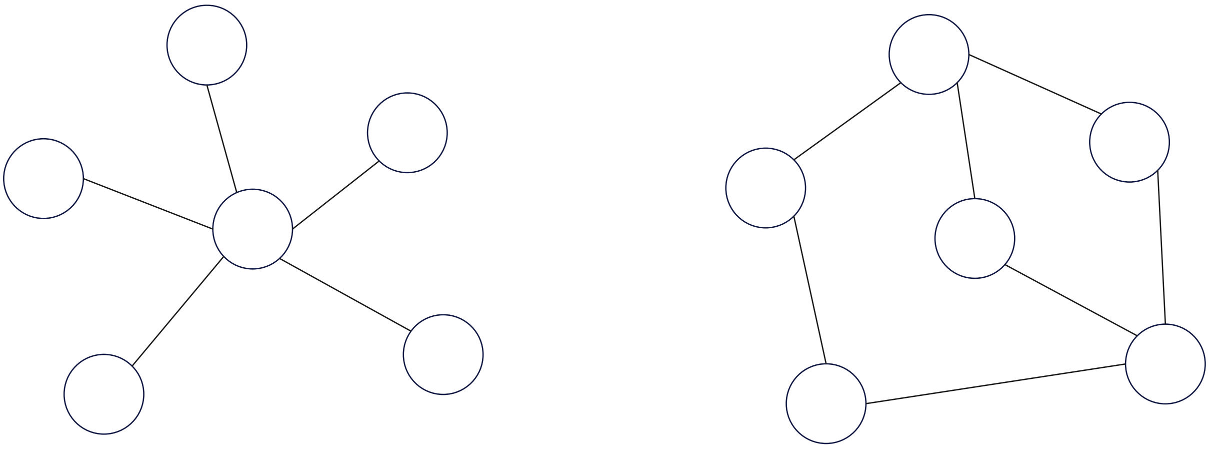

Time delay (communication + cloud computing)
Time delay (edge computing + communication + cloud computing)
$T_{Edge\ computing}$
$T_{communication}$
$T_{cloud\ computing}$

Figure 17 Edge computing reduces time delay significantly when $T_{communication}$ dominates

4.2.1 Fault tolerance and topology

Fault tolerance is the property that enables a system to continue operating correctly in the event of a failure due to one or more faults of its components. The fault-tolerant capability allows DT to continue its intended services, possibly at a reduced level, rather than failing when some part of the system fails [212]. Currently, most bridge DTs offer data storage and analysis on the cloud. Still, the services will fail when cloud servers become unresponsive, such as temporarily losing communication. Suppose it occurs when a disaster is happening or is predicted to happen. In that case, it might be fatal to the physical bridge and the users on-site, so developing a bridge DT with the required fault-tolerant capability is necessary. Edge computing and AIoT can also help to enhance the current bridge DT system resilience. For example, even if communication between the edge and cloud breaks down, the edge devices can still make an AI-based decision to take an appropriate and timely measure via bridge or transport control systems as a response, e.g., load restriction, traffic diversion, or even closure. Depending on task complexity, such edge computing can be designed on base stations, gateways, embedded systems, or sensor nodes. The closer to the physical bridge, the more resilient the design will be.

Besides, the communication topology is also significant to system resilience. A decentralized mesh network (even partial mesh) has better fault tolerance than a centralized star network, as shown in Figure 18. As presented in 2.2.2, non-cellular LPWAN is suitable for remote bridge management thanks to its decentralized transmission mode and long-distance coverage (over 10 km in rural areas), such as LoRa. Therefore, non-cellular LPWAN with a mesh network can be utilized for local communication on the site of physical bridges and integrated into the cloud-based DT architecture to enable fault tolerance for a temporary loss of communication. Meanwhile, by combining AI-based edge computing, the edge devices and control system of physical bridges can perform as a self-adaptive subsystem when the cloud servers become unresponsive, and their performance can be simulated and predicted at the cloud level. Hence, the whole bridge DT can continue to work at a reduced level without complete failure. Finally, the physical bridge and digital-twin bridge can re-synchronize after the restoration of communication.



(a) Cellular network

(b) Non-cellular LPWAN

Figure 18 (a) Cellular network based on star network (b) Non-cellular LPWAN based on the mesh network

4.2.2 Mathematical idealization

A physical bridge, i.e., PE, can be described as a discrete system in Eq. 6. x_t and x_{t+1} are the states of PE at time t and $t+1$. u_t is the input variable at time t. y_t is the observed variable from IoT sensors. e_t is the error of the measurement.

$$x_{t+1} = f(x_t, u_t), \quad y_t \ = h(x_t, u_t, e_t) \tag{6}$$

Here, $\mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_e} \rightarrow \mathbb{R}^{n_y}$. The edge-computing model was trained with the observed variable $y \in \mathbb{R}^{n_y}$, so the inference can be represented as $v_t = g(y_t)$. v_t is the result to support decision-making, i.e., to calculate the input variable u_{t+1} at time $t+1$. Therefore, $u_{t+1} = d(u_t, v_t)$.

If such computing is taken on the cloud, i.e., x_t, y_t and u_t are transmitted to VE, it can leverage powerful computational capability and massive multi-source data ($\varphi \in$

$\mathbb{R}^{n_\varphi}$) in the cloud, which can help to make more precise and holistic decisions. Therefore, there is $V_t = G(y_t, \varphi_t)$.

However, it will also bring in a time delay T_{delay} between edge and cloud, i.e., PE and VE. Given bi-directional time delay in both uplink and downlink, the input variable u derived from cloud computing has a lag of $2d$ behind the result based on edge computing, i.e., $u_{t+2d} = D(u_t, V_t, \theta)$. $\theta \in \mathbb{R}^{n_\theta}$ is the set of model parameters for prediction to offset the time delay. d is the ratio of T_{delay} to the measurement period T_{period}. Therefore, considering the time delay between PE and VE, the cloud-based bridge system can be represented as Eq. 7.

$$x_{t+2d+1} = F(x_t, u_{t+2d}), \quad y_{t+2d} = h(x_{t+2d}, u_{t+2d}, e_{t+2d}) \tag{7}$$

Therefore, edge computing is suggested for quick analysis and response due to its low latency, such as under emergent circumstances. Moreover, edge-based algorithms also pursue low computational complexity in practice, e.g., $u_{t+1} = s(v_t)$ in edge computing is usually based on straightforward “what-if” analysis or fuzzy control rules. In contrast, cloud computing is recommended for holistic decision-making with big-data analysis from multiple sources to achieve a long-term maintenance strategy for the bridge.

For one-way communication, i.e., sending all three variables x_t, y_t, u_t to the cloud, the communication complexity $D(F) = O(x) + O(y) + O(u)$, where the variables have x bits, y bits, and u bits respectively. Most of the time, the communication time in and among edge devices on a single physical bridge can be negligible.

Furthermore, the total computational resources required by tasks running on the edge device should not exceed its processing capability [26].

$$C_{edge}^{processor} > \sum task_i^{processor} \tag{8}$$

$$C_{edge}^{RAM} > \sum task_i^{RAM} \tag{9}$$

4.2.3 Petri-net modelling

The bridge DT can be taken as a discrete event dynamic system with distributed architecture. Therefore, it can be described with a Petri net (PN), i.e., a bipartite, weighted, and directed graph (digraph) that has been proven to be an efficient tool for the modelling, analysis, and control of a discrete event system (DES) [213]. A stochastic time PN can help demonstrate system resilience and time delay and trace data flow and sources in the system for DT services.

A PN comprises two types of nodes, i.e., places and transitions, with arc connections from a place to a transition (or a transition to a place). In the graphical representation, a place $p \in P$ is drawn as a circle representing a condition (a particular state of the system). In contrast, a transition $t \in T$ is marked as a box describing an event (dynamic activity). Arcs are labelled with the corresponding weights $w \in \mathrm{W}$ ($w = 1$ by default). Places are visited by k tokens, representing data items moving through a PN, like communication packages. Marking M is given by a vector that refers to the distribution of tokens throughout all the places at a specific time, indicating the state of the PN. The firing rules indicate when and how tokens are created and destroyed in a new marking M'. More details about Petri nets can be found in [214], [215].

Mathematically, a stochastic time PN can be described as below:

$$N = (P, T, A, \mathrm{W}, \Lambda, \Theta\ \Theta) \tag{8}$$

Where P is the finite set of places, $P \neq \phi$; T is the finite set of transitions, $T \neq \phi$; $A \subseteq (P \times T) \cup (T \times P)$ is the set of arcs; $W: A \rightarrow \{1,2,3\ldots\}$ is the weight function on the arcs; $\lambda \in \mathit{\Lambda}$ is the firing rates associated with transition, which are related to probabilities of successful communication; $\theta \in \mathit{\Theta}$ is the time elapsed in state transition.

Here, the firing rules are summarized as follows:

1. Transition t_i consume the tokens from each available input arc and generate $\omega_{i,output}$ tokens at each output arc. $\omega_{i,output}$ is the weight on the output arc.
2. Transition t_i is enabled if the input place p_j has at least $\omega_{j,input}$ tokens. $\omega_{j,input}$ is the weight on the input arc.
3. An enabled transition t_i fires according to the firing rate λ_i with probability R, which can be expressed as below:

   $$t_i = \begin{cases} 1 & (\text{R}=\lambda_i) \\ 0 & (\text{R}=1-\lambda_i) \end{cases}$$

4. After firing, transition t_i removes all the tokens in the place p_i and add $w_{i,output} \cdot t_i$ tokens into the next place.

The proposed framework is modelled in a PN, which has seven places ($p_1, p_2, \ldots p_7$) and six transitions ($t_1, t_2, \ldots t_6$), as shown in Figure 19. The starting place p_1 holds the token of the observed variable y from PE, i.e., physical bridge. t_1 and t_5 are taken on edge devices, such as sensor nodes, embedded systems, and gateways. t_2 and t_4 are communication between PE and VE and fired according to the firing rates λ_2 and λ_4. t_3 is taken on cloud servers. p_4 holds the token of the variable v collected from multiple resources in the cloud. Finally, the decision-making variables from edge computing and (or) cloud computing converge at p_7 for the control or adjustment of PE to generate a new token of the observed variable y'.

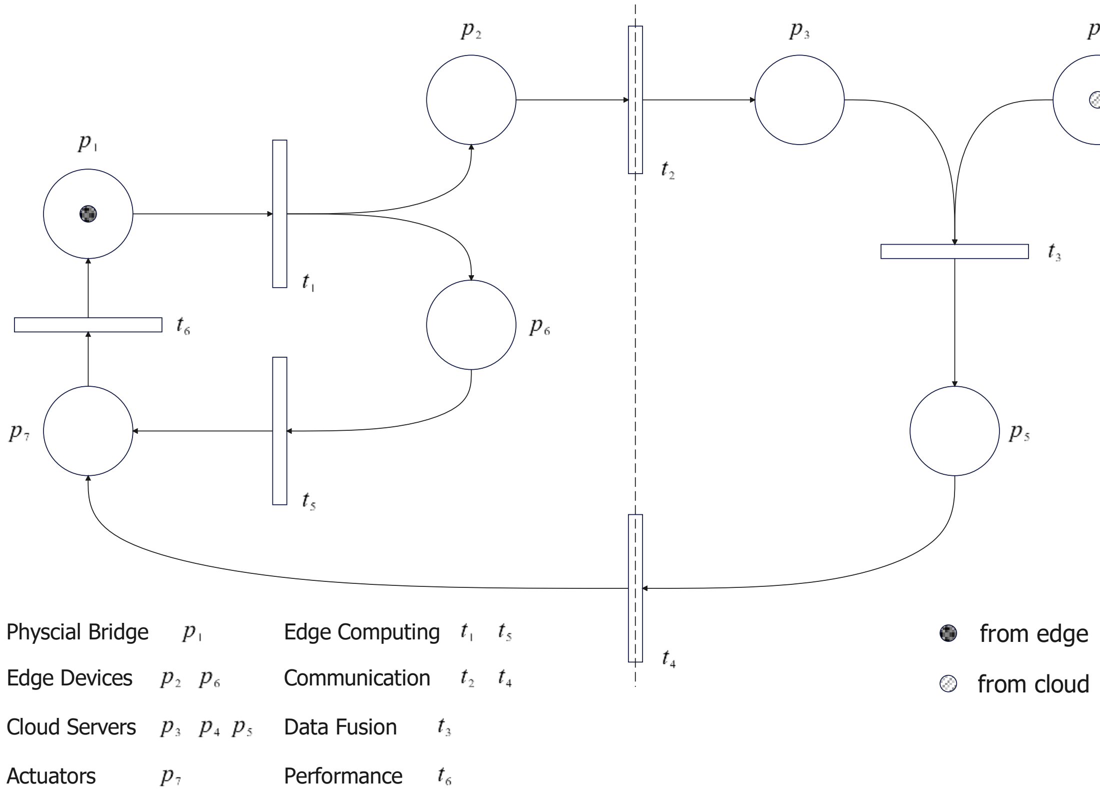


Figure 19 Petri-net modelling for a bridge DT system initialized from M_0

As can be seen from the PN, there is an edge-based loop $t_1 \rightarrow t_5$ and a cloud-based loop $t_1 \rightarrow t_2 \rightarrow t_3 \rightarrow t_4$. Here, assume the edge loop working as E and failure as E', and so does the cloud loop, i.e., C and C'. Therefore, the edge loop working probability is $R(E) = \lambda_5$ and the cloud loop working probability is $R(C) = \lambda_2 \cdot \lambda_3 \cdot \lambda_4$.

1) If both loops (after t_1) are mutually independent, the results can be divided into three categories: a) the result is determined with both edge and cloud loops, wherein the probability is $R(E) \cdot R(C)$; b) determined with either edge or cloud loop, wherein the probability is $R(E') \cdot R(C)$ or $R(E) \cdot R(C')$, respectively; c) system failure, wherein the probability is $R(E') \cdot R(C')$. Because of $0 < \lambda_i < 1$, $R(C') - R(E') \cdot R(C') = \lambda_5 \cdot (1 - \lambda_2 \cdot \lambda_3 \cdot \lambda_4) > 0$. Therefore, the system failure probability decreases by adding the edge loop. Moreover, to achieve a significant enhancement of system robustness, λ_5 should be much greater than $\lambda_2 \cdot \lambda_3 \cdot \lambda_4$.

2) If both loops (after t_1) are mutually exclusive with an extra rule that t_5 only fires when t_2 or t_3 or t_4 does not fire, i.e., the edge loop is only enabled when the cloud loop fails. Thus, according to conditional probability, there is $R(C') - R(E'|C') = R(E|C') = (1 - \lambda_2 \cdot \lambda_3 \cdot \lambda_4) \cdot \lambda_5 > 0$. Therefore, the system becomes more robust by adding the edge loop.

3) Moreover, the total time elapsed in a path can be calculated by summing all the elapsed times in the firing schedule [215]. For the edge loop, the time elapsed $\theta_{edge} = \theta_1 + \theta_5$, while for the cloud-based loop, the time elapsed $\theta_{cloud} = \theta_1 + \theta_2 + \theta_3 + \theta_4$. Hence, the difference is $\Delta = \theta_2 + \theta_3 + \theta_4 - \theta_5$. Here, θ_2 and θ_4 are communication time elapsed, which θ_3 and θ_5 are computational time at the cloud and the edge. Hence, a trade-off between computing and communication complexity is required to guarantee the system's performance.

4.3 Framework design and development

4.3.1 AI-based edge computing

When agent-based drones, robots, and versatile sensor networks are leveraged for bridge regular inspection and real-time monitoring, AI-based edge computing can enable them with autonomous capability for preliminary analysis and decision-making, such as damage detection, bridge assessment, and early warnings. An advantage in this way is that edge computing can reduce data complexity significantly by converting them to advanced information or knowledge according to information hierarchy (i.e., DIKW), as shown in Figure 20. Its ultra-lightweight data/information streaming can enable bridge DT to satisfy restricted communication requirements, such as LPWAN.

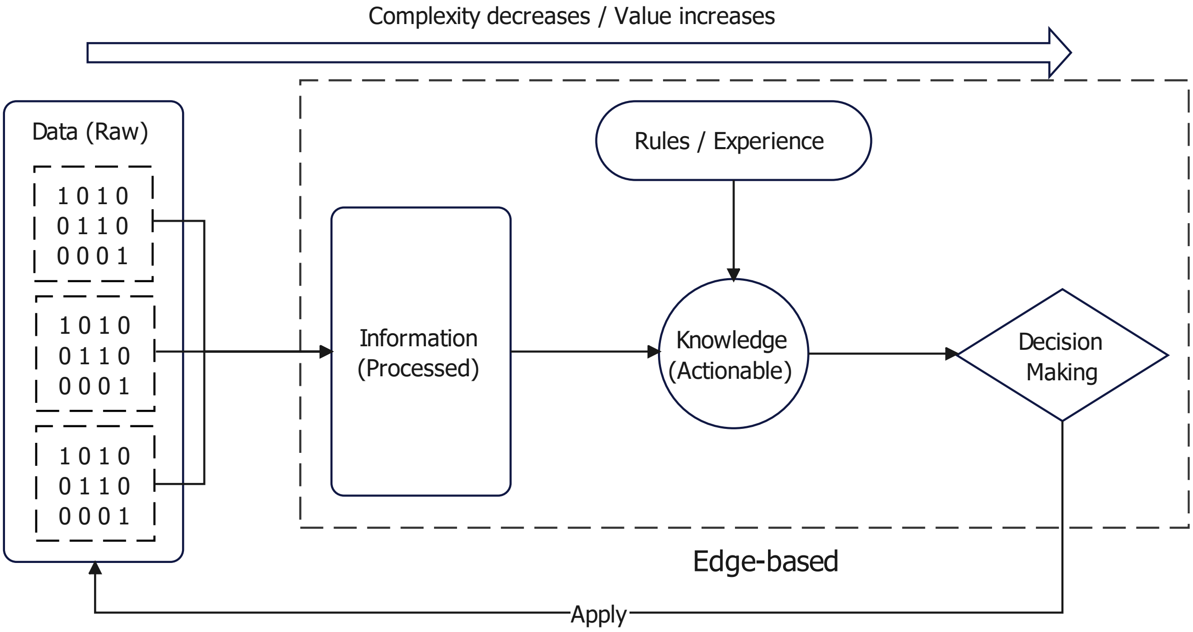


Figure 20 The edge loop according to DIKW to reduce complexity

The derived information or knowledge can be transmitted to cloud servers to achieve DT services directly (such as visualization) or participate in the bipartite interactive computing for function f (see Eq. 5) based on different data sources owned by edge and cloud. It can significantly reduce the delay of cloud-based DT services using appropriate sequential design (see Figure 17). For example, in the drone-enabled bridge inspection, the extracted semantic information based on DL and computer vision, such as defect location and severity, can be synchronized to the bridge DT quickly and conveniently through LPWAN instead of massive image and point-cloud data. This method can enhance maintenance efficiency through DT services such as historical query, big-data analysis for defect causes, and optimization for inspection and repair.

Another advantage is that preliminary decision-making based on edge computing can provide the quickest response and still perform the task even if cloud servers become unavailable. For example, the detected bridge damage from a sensor network, which has influenced bridge serviceability, can trigger weight restriction or even closure through actuators and monitors immediately. Moreover, it can also transmit the derived information to cloud servers or adjacent bridges for collaboration, such as traffic diversion.

Notably, edge computing can be taken on different roles in architecture, such as sensor nodes or gateways, as well as various edge devices equipped with robotic agents or attached to physical bridges, including field-programmable gate array (FPGA), microcontrollers, single-board computers, etc. The power can be supported by batteries, sustainable power supply, and energy harvest [216]. The edge-computing tasks and power restriction should not exceed the device's computational capability (see Eq. 8 and Eq. 9). Meanwhile, the data owned by the edge is limited to the local physical bridge and environment, so edge computing only aims to provide preliminary analysis and decision-making. Moreover, to achieve high resilience, the edge device for decision-making should usually be placed as close to physical bridges as possible (explained in 3.2). Still, it can also be designed for hierarchical architecture for different-level tasks, which will be discussed in the next section.

4.3.2 LPWAN communication

As reviewed in 2.2.2, LPWAN is a set of low-power, long-distance communication technologies. As claimed [56]–[58], [217], [218], they can sustain a long-term work of up to several years for battery-operated sensor nodes and over 10 km coverage in rural areas, which is beneficial to the connection for bridges in resources-constraint environments. However, LPWAN is restricted by its low data rates, duty cycle, etc., so it cannot transmit massive heterogeneous data in regular inspection and real-time monitoring, e.g., images and point clouds. Thus, AI-based edge computing is designed to reduce complexity significantly by converting data to advanced information or knowledge, according to DIKW. The derived information can be organized in a code system with a predefined protocol [219]. Moreover, appropriate compression methods can also be applied before transmission to achieve more efficient transmission, such as run-length encoding (RLE), a lossless compression approach especially suitable for binary images.

Moreover, non-cellular LPWAN does not require commercial base stations and can take advantage of free ISM bands, providing flexible and low-cost communication to create bridge DTs. It can be utilized in a hierarchical communication architecture designed for different-level tasks, wherein sensor nodes, gateways, and cloud servers are organized from bottom to top. Gateways can simultaneously manage multiple sensor nodes and embedded systems (e.g., multiple bridges and robotic agents) in an area through LPWAN. They can communicate with cloud servers as well as with each other. Thus, bridge DTs are enabled with fault tolerance (or resilience) by combining edge-based AI for normal functions when cloud servers become unavailable temporarily. Furthermore, sensor nodes are designed to communicate with gateways and connect to each other via non-cellular LPWAN, which can guarantee preliminary decision-making and quick response on physical bridges even if gateways break down (resilience). In the prototype, LoRa is selected for DT communication to support bridge O&M. The complete hierarchical communication architecture is shown in Figure 21.

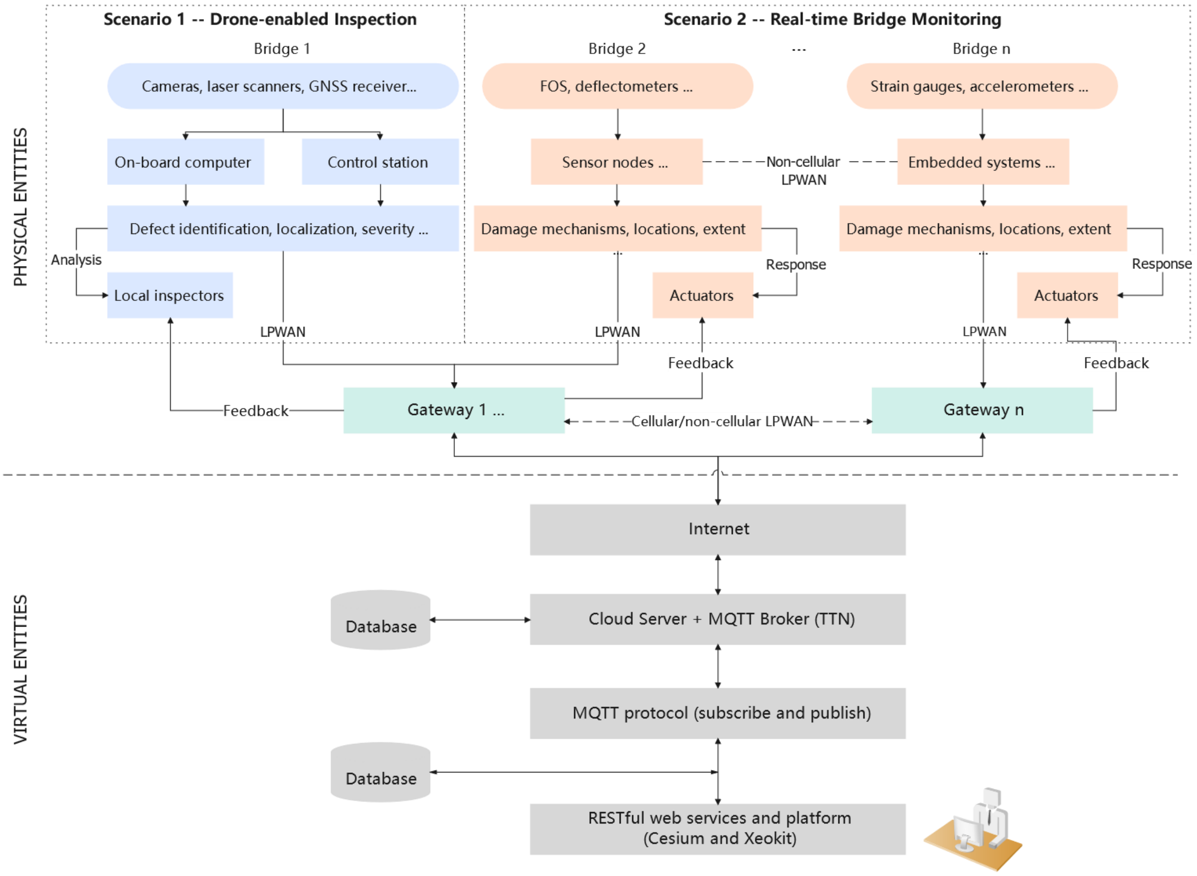

PHYSICAL ENTITIES
Scenario 1 -- Drone-enabled Inspection
Bridge 1
Cameras, laser scanners, GNSS receiver...
On-board computer
Control station
Defect identification, localization, severity ...
Analysis
Local inspectors
LPWAN
Scenario 2 -- Real-time Bridge Monitoring
Bridge 2
...
Bridge n
FOS, deflectometers ...
Strain gauges, accelerometers ...
Sensor nodes ...
Non-cellular LPWAN
Embedded systems ...
Damage mechanisms, locations, extent
Damage mechanisms, locations, extent
Response
Response
Actuators
Actuators
LPWAN
LPWAN
Feedback
Feedback
Feedback
Gateway 1 ...
Cellular/non-cellular LPWAN
Gateway n
VIRTUAL ENTITIES
Internet
Database
Cloud Server + MQTT Broker (TTN)
MQTT protocol (subscribe and publish)
Database
RESTful web services and platform (Cesium and Xeokit)

Figure 21 Proposed framework and developed prototype for bridge DT

4.3.3 Cloud and protocols

Cloud servers are designed to provide a CDE for data from multiple sources, including edge-based data from regular inspection and real-time monitoring and cloud-based data such as design and construction documents, historical records, inventory, traffic, weather, and disasters. Relational and non-relational databases, like MySQL and MongoDB, can store and manage such heterogeneous data and information. Meanwhile, cloud computing is another crucial function. The huge and sophisticated models for complex DT services, e.g., structural analysis, DL, multi-objective optimization, and holistic decision-making, which require powerful computational capability, are deployed on cloud servers within dockers. The models and data can be conditionally accessed and modified by different stakeholders. The results can be sent back to end devices on physical bridges through the downlink of LPWAN for performance.

The proposed information or knowledge from cloud servers can be transmitted to HMI through HTTP or MQTT protocols. In the prototype, the TTN (The Things Network) server and the desktops in the lab are integrated as cloud servers to support the web-based platform. The bidirectional communication among cloud servers and the interface is achieved with MQTT protocol thanks to its lightweight and fast transmission capability. Specifically, it can be implemented with "subscribe" and "publish" between brokers and clients using Eclipse Mosquito or Node-RED, as shown in Figure 22.

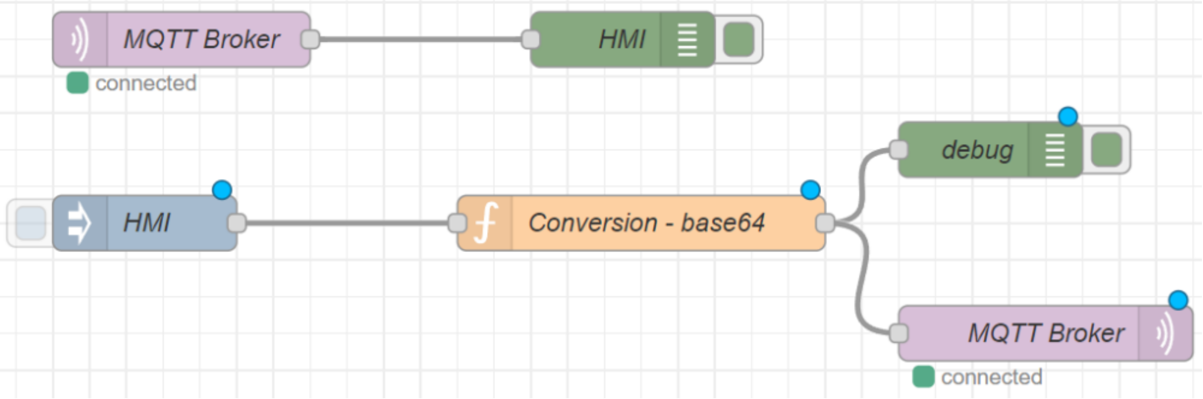


Figure 22 MQTT implementation between cloud servers and HMI

4.3.4 Web-based platform

The bridge DT platform is designed to perform through web services based on the RESTful framework. It has a user-friendly interface, which can enable users to access the available information according to their permissions. In the prototype, Cesium and Xeokit are employed in the platform for GIS and BIM, respectively. Node.js and npm-anywhere (i.e., a static file server) support the web-based interface. The bridge location, traffic conditions, ambient situation (weather and tides), and project description are displayed in the Cesium, wherein 3D tiles are utilized for bridge visualization. Xeokit enables users to manipulate each element of the BIM model based on the IFC file. Then, the bridge condition, such as defect location and severity, bridge serviceability, and structural assessment, are displayed. The web-based click event can trigger various DT services, such as query, knowledge-based reasoning, and maintenance planning. Such a platform can enable users to interact with entities and nodes in VE and PE, e.g., to reconfigure and reboot the edge device in the VE remotely from the platform if the device is registered with a fault.

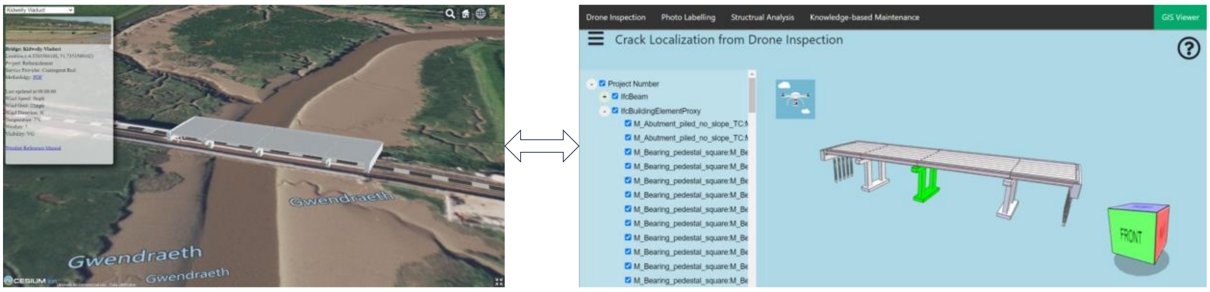


Figure 23 Platform across Cesium (GIS part) and Xeokit (BIM part)

Finally, following the above sections, the proposed AIoT-informed framework and the developed cross-platform prototype are shown in Figure 23. The framework and prototype have high efficiency and low latency, achieved through AI-based edge computing. Moreover, edge computing and the resilient hierarchical communication architecture with cellular and non-cellular LPWAN can enable the PEs (i.e., physical bridges) with the capability of preliminary data analysis, decision-making, and quick response, even under the temporary loss of communication, to guarantee the system fault tolerance. Meanwhile, the PEs with edge devices can perform as a self-adaptive subsystem when cloud servers become unavailable. Their performance can be predicted by cloud-based VE using simulation or ML. Then, PE and VE will synchronize again after communication recovery.

4.4 Proof of concept

Firstly, the proposed framework is validated under three cases during bridge O&M to demonstrate its functionality, i.e., 1) drone-enabled bridge inspection, 2) vibration-based bridge monitoring, and 3) dynamic evacuation when cloud servers become unresponsive. Then, a comparative analysis between the proposed framework and the previous bridge DTs is conducted.

4.4.1 Drone-enabled bridge inspection

The experiment aims to synchronize the sufficient defect characteristics to the bridge DT during drone flight in a communication-constraint environment (e.g., LoRa) to update the bending stiffness (BS) reduction coefficient for structural assessment instead of bringing massive inspection images to the office (like in the conventional workflow) or to another place under excellent communication for synchronization. This method enables on-site inspection and back-end DT services to collaborate simultaneously, e.g., structural assessment, historical query, in-depth inspection, mechanism analysis, and even instant repair, enhancing maintenance project efficiency.

The AI-based processing is taken on a Raspberry Pi 4 Model B, which can be utilized as a drone on-board computer or a controller in situ. As a prerequisite, the drone needs calibration before inspection using a chessboard at different distances and angles to obtain the proportional scales between the image pixel and actual size. Deep convolutional neural networks (DCNN) can enable drones with automatic defect-detection ability, which has been widely accepted and commonly used in image-based defect detection. Here, a dataset created for bridge crack detection [220] is selected for the experiment, including the 2011 background and 4058 crack images (224×224). The images are resized to 32×32 and trained with a simplified LeNet-5 (for acceleration of the on-board computing process) through TensorFlow on the Google Codelab under the configurations: train-validation-test split – 60%:20%:20%; optimizer – stochastic gradient descent (SGD); learning rate – 0.001; batch size – 128. The training process and model performance in the test set are shown in Figure 24 and Table 4. Then, the model is converted into a specific version for tinyML through TensorFlow Lite, which is especially suitable for the DL model deployment on microcontrollers and embedded systems with improved efficiency. The crack identification for one image is less than 1 second in the experiment.

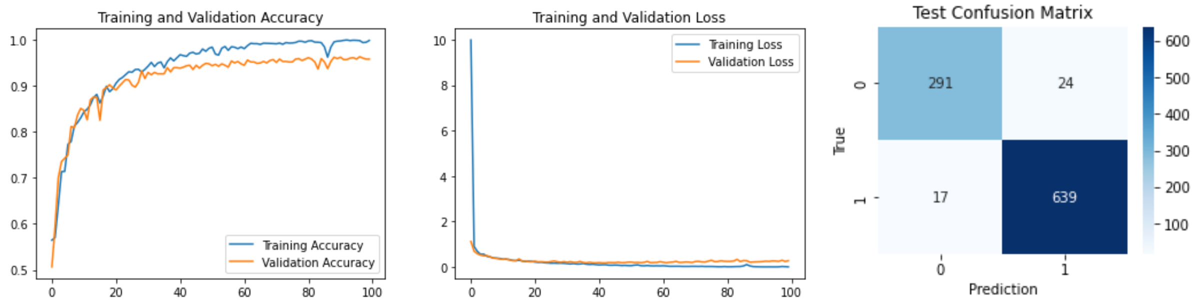
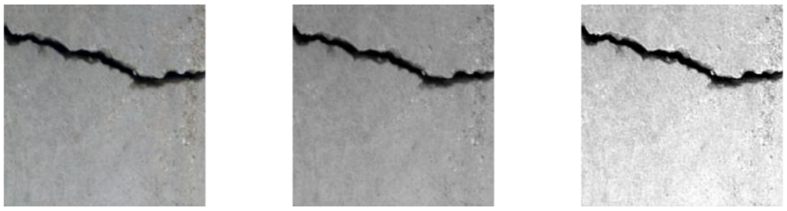


Figure 24 Training and test for crack detection

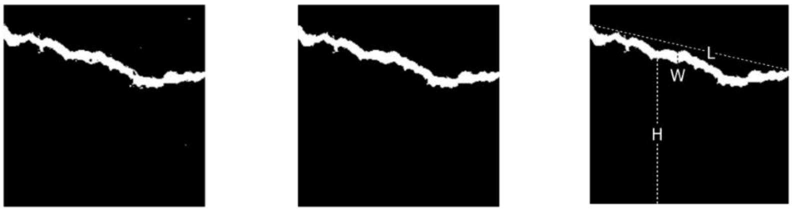
Table 4 Model performance evaluation

Model	Accuracy	Precision	Recall	F1 score
LeNet-5	0.95	0.95	0.91	0.93

Subsequently, detected crack images are segmented into binary images with background and crack through image processing (OTSU thresholding and morphological operations). Then, the crack characteristics can be calculated statistically, such as H_{crack}, W_{max} and L, as shown in Figure 25, by combining the distance from the lens to the objective surface (measured with an ultrasonic ranger or a laser scanner, etc.)



Original Crack   Gray-scale   imadjust + imsharpen



Binary image with Otsu   Remove areas less than 10   Crack characteristics

Figure 25 Crack segmentation through image processing

Crack orientation can be determined with the camera angle and the flight attitude from IMUs, such as transverse or longitudinal. Furthermore, the defect (e.g., crack) can be localized in the bridge coordinate by combining GNSS positioning (such as RTK or PPK), IMUs, and distance ranger during flight, as shown in Figure 26 [53]. Given the situation without stable GNSS signals, computer vision (such as bridge element recognition), IMUs, and distance rangers can be leveraged for drone positioning and defect localization through the Kalman Filter.

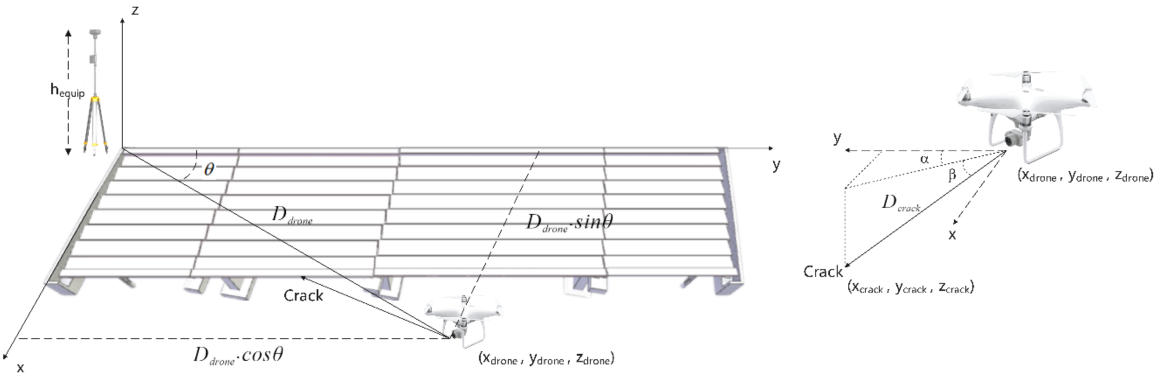

z
h_equip
θ
D_drone
D_drone·sinθ
y
Crack
x
D_drone·cosθ
(x_drone, y_drone, z_drone)
y
α
β
D_crack
(x_drone, y_drone, z_drone)
x
Crack
(x_crack, y_crack, z_crack)

Figure 26 PPK approach for defect localization in the bridge coordinate system

Given the earth's ground can be taken as a plane within just a few-kilometre distance, D_{drone} can be calculated as Eq. 9. The defect coordinates can be calculated as Eq.10 -12. Then, the coordinates can be further linked to the precise bridge element, such as the beam, deck, and pier, according to geometric information.

$$D_{drone} = R\sqrt{(\phi_2 - \phi_1)^2 + (\lambda_2 - \lambda_1)^2} \tag{9}$$

Where L – horizontal distance; R – earth's radius (the parameter that needs to be calibrated); φ_1, φ_2 – base and drone latitude; λ_1, λ_2 – base and drone longitude.

$$X_{crack} = D_{drone} \sin\theta - D_{crack} \cos\beta \cos\alpha \tag{10}$$

$$Y_{crack} = D_{drone} \cos\theta - D_{crack} \cos\beta \sin\alpha \tag{11}$$

$$Z_{crack} = H_{drone} - H_{base} + h_{equip} - D_{crack} \sin\beta \tag{12}$$

The AI-derived critical information for structural assessment, i.e., defect types, characteristics, coordinates, and orientation, can be encoded as Figure 27 with three significant digits (i.e., a single float) for transmission. It can satisfy the strictest payload requirement of LoRa (at SF12/125kHz) with the maximum range in EU868. The calculated airtime is 2.138 seconds, which can achieve synchronization in near real-time.

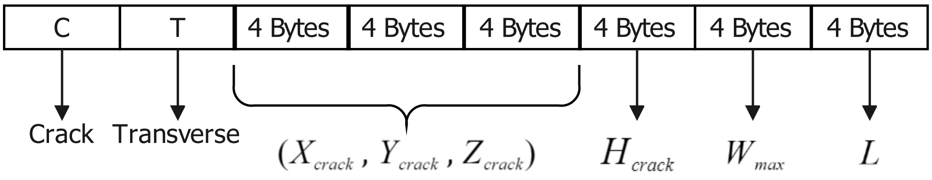


Figure 27 Encoded defect information to synchronize for structural assessment

BS reduction coefficient β is calculated as $\beta = K_i/K_1$, where K_1 is the initial stiffness of the beam in the elastic stage and K_i is the ith loading. The relationship of β to W_{max} and H_{crack} can be derived from the test on the specimen as Eq. [221]. Therefore, β can be calculated using the synchronized crack information so that the BS reduction can be assessed successfully.

$$\beta = \{f(W_{max}),\ g(H_{crack}/h)\}_{min} \tag{13}$$

Where h is the beam height.

Sometimes, defect profiles are unusual, such as the crack shown in Figure 28. The segmented crack profile can be losslessly compressed via RLE significantly, i.e., 8048 to 654 bytes, and then completely recovered in the cloud server. This approach performs better than the previous research for image transmission through LoRa based on lossy compression [222], [223] and can still satisfy the DT services, such as visualization and evaluation. However, it is worth noting that the LoRa data rate is still relatively low. Even if the example image only requires three communication packets at SF7/125KHz, it still needs an airtime of 1.107 seconds and a duty cycle of 73.8

seconds for such a transmission. In this case, other LPWAN technologies with higher bandwidth, such as NB-IoT, are more recommended.

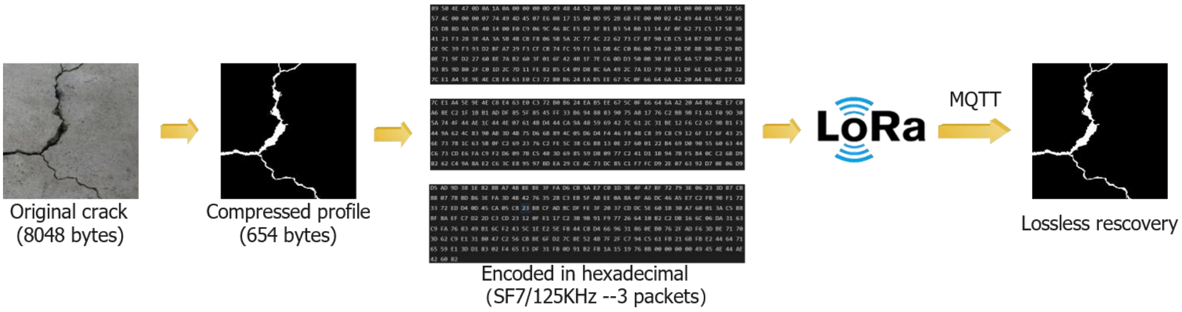


Figure 28 Synchronization of crack profile through LoRa and RLE

This experiment demonstrates that the proposed framework with the developed prototype can synchronize the drone-enabled inspection to the cloud server in near real-time for bridge DT services, such as structural assessment and visualization. Meanwhile, the feedback can be transmitted to local inspectors or agent-based drones through the on-board computer or the drone controller. The complete procedure can be shown in Figure 29.

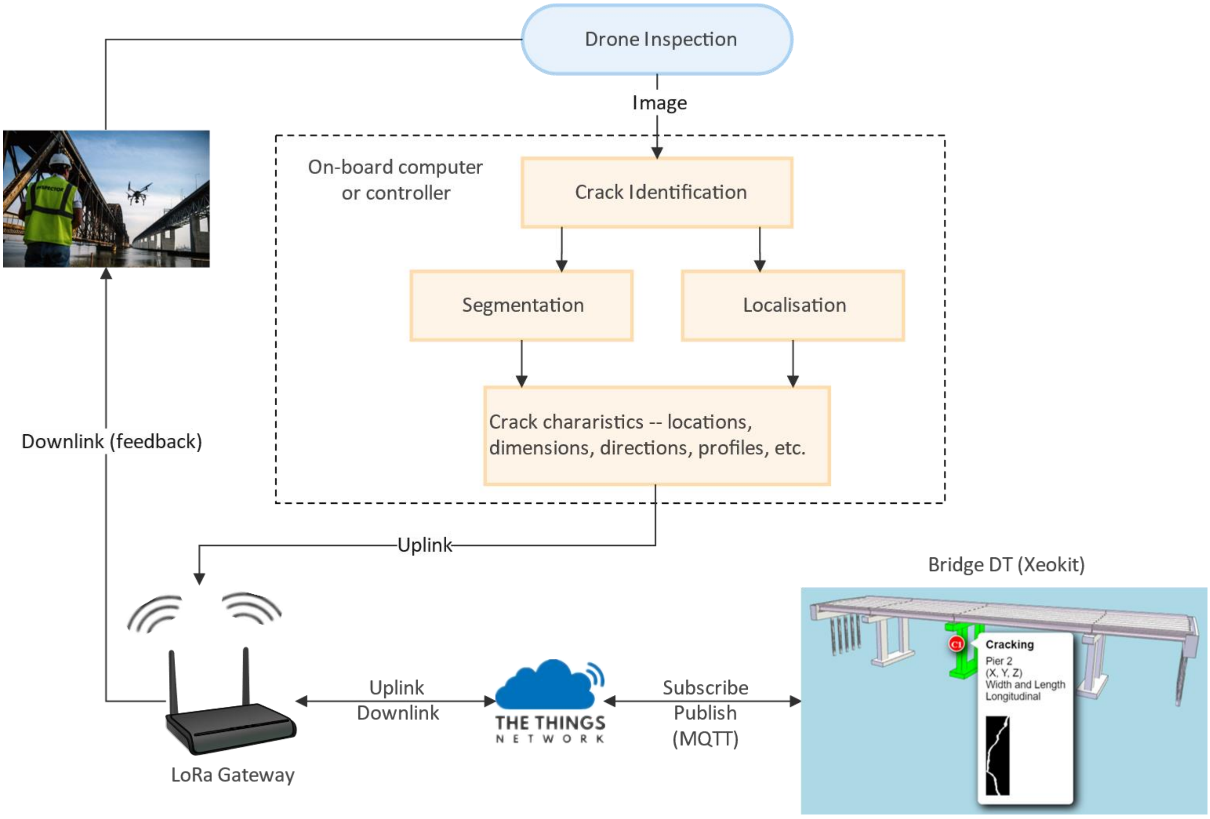


Figure 29 Developed bridge DT prototype for drone-enabled bridge inspection

4.4.2 Vibration-based monitoring

Previous research [26] developed a cloud-based bridge DT to achieve real-time SHM based on vibration signals using a pre-trained surrogate model based on DL. Still, it relies on excellent communication (i.e., 5G), and its services will fail when cloud servers become unresponsive (lack of resilience). This experiment aims to achieve similar real-time SHM in the prototype and demonstrates the fault tolerance of the proposed framework for temporary loss of communication between edge and cloud.

The public dataset of acceleration signals from the VBM project of the KW51 bridge [80] is employed in the experiment, generated from 6 uniaxial accelerometers during train passes before and after bridge repair (i.e., the damaged and healthy condition, respectively). Its sampling frequency is 825.8Hz, and the resolution is 24-bit. Such vast data is a challenge for synchronization, especially in a communication-constraint environment, i.e., LoRa here. Edge computing is taken on the Raspberry Pi 4 model

B. Two different machine-learning approaches are developed for bridge risk identification. One is based on the support vector machine (SVM) with hand-crafted features, including signal features [224] or wavelet-packet energy (WPE) as Eq. 14, and the other is based on 1D-CNN with multi-channel input (shown as Figure 30), which is similar to the DNN model used in previous bridge DT [26].

$$WPE_{level,i} = \sum |x(n)|^2 \tag{14}$$

Note: wavelet-packet energy (WPE) of discrete-time signal for each node (level = 3, i = 1 ~ 7).

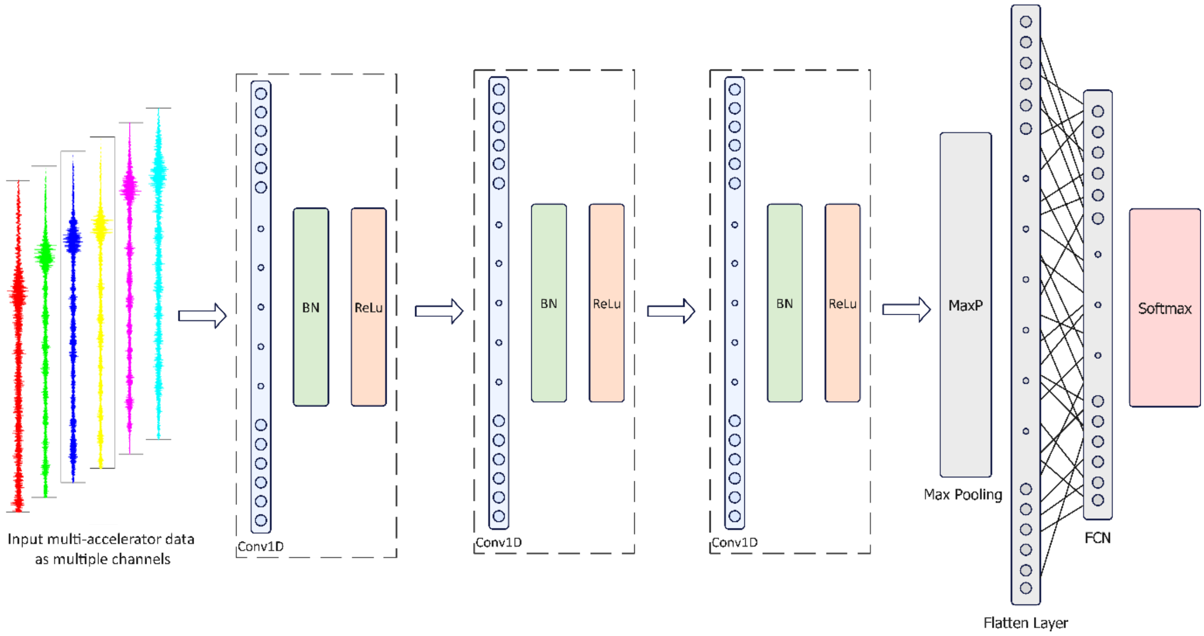


Figure 30 1D-CNN architecture with multi-channel input utilized for pattern recognition

The SVM model with the hand-crafted features is trained on a laptop using GridSearchCV to find the optimal parameter values. Its confusion matrices on the test set are shown in Figure 31 and Table 5. The 1D-CNN model with raw data (50176×6) is trained through TensorFlow on the Google Codelab, i.e., train-validate-test split – 60%:20%:20%; optimizer – SGD; learning rate – 0.001; batch size – 64. Then, it can be converted to a tinyML version and deployed on the Raspberry Pi (i.e., the embedded system installed on the physical bridge). Two LED lights (green and red) are connected to the GPIO pins of the Raspberry Pi, and the result can be displayed by switching either of them on. With normal communication, the edge can transmit the identified bridge pattern and the timestamp to the cloud using the predefined code system and then receive cloud-based feedback. Otherwise, when communication between the edge and cloud is lost, the local embedded system can perform preliminary pattern recognition and trigger corresponding measures on the physical bridge, such as weight restriction, traffic diversion, or even closure. The edge-based inference of SVM with signal features or WPE takes an average time of 1.2104 and 1.5309 seconds on the test set, while 1D-CNN with raw data takes 0.7825 seconds. Meanwhile, 1D-CNN has the best performance, as shown in Table 5.

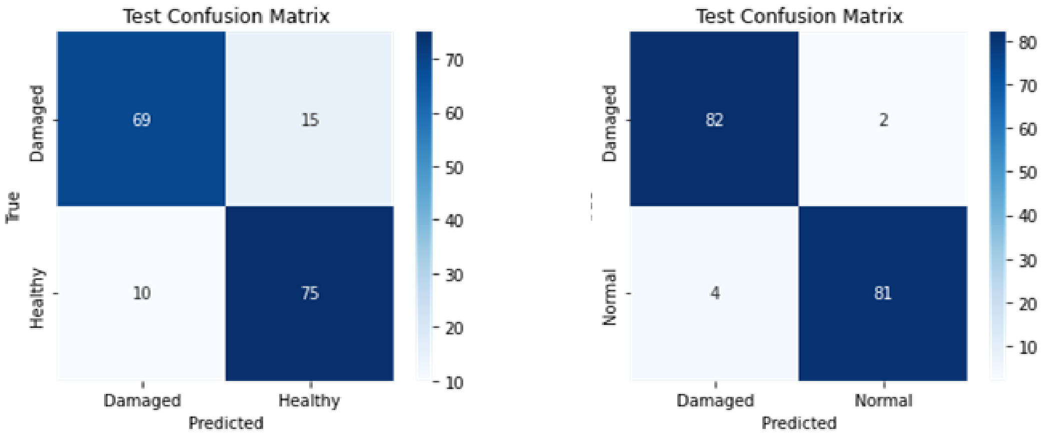


Figure 31 Test confusion matrices for SVM with signal features and SVM with WPE

Table 5 Model performance for pattern recognition

Model	**Input**	**Accuracy**	**Precision**	**Recall**	**F1 score**
SVM	SF	0.85	0.83	0.88	0.86
SVM	WPE	0.96	0.98	0.95	0.96
1D-CNN	Acceleration	1	1	1	1

Note: SF – signal features; WPE – wavelet-packet energy (level-3); acceleration data – 50176×6.

It is worth noting that the derived signal features and WPE can be either utilized at the edge or transmitted to the cloud, e.g., signal features require a total of 33 bytes with the calculated airtime of 2.302 seconds at SF12/125kHz for LoRa because their communication complexity has become much lower than the transmission of raw data in the previous DT. This experiment demonstrates that the proposed framework with the developed prototype can achieve similar performance to the previous research [26] for real-time SHM but does not rely on excellent communication anymore and has better fault tolerance in operation to guarantee the safety of the physical bridge. The complete procedure is shown in Figure 32.

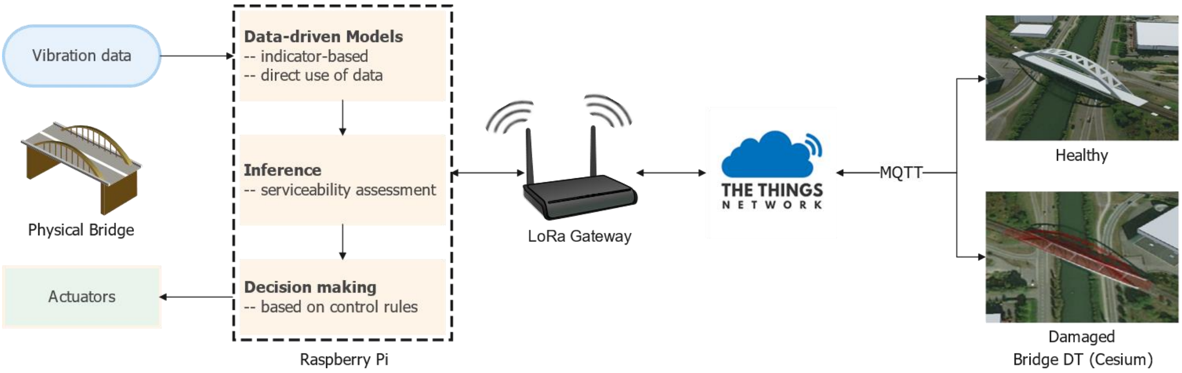


Figure 32 Proposed framework and developed prototype for bridge VBM

4.4.3 Dynamic evacuation

As an emergency response, dynamic evacuation with route planning is necessary when a disaster is happening or is predicted to happen on the site of physical bridges, such as flash floods or earthquakes. Route planning is a well-known problem which can be solved with many approaches, such as Floyd [225] and Dijkstra [226] algorithms. However, under extreme weather conditions, the cloud server likely becomes unresponsive, e.g., the internet or gateways break down temporarily, resulting in DT service failure for dynamic route planning in the cloud-based system. It is dangerous for public users travelling on the site of bridges. This experiment aims to demonstrate the resilience of the proposed framework to endure a temporary loss of communication at different levels.

An open-source emulator for the LoRa network [227] is employed for the experiment. Suppose there is an area with multiple bridges under the threat of flash flooding, as shown in Figure 33. People must evacuate from the left side (flooding area) to the right side (safe area) of the dashed line in the bridge network. Gateways activate the LoRa sensor nodes for water-level monitoring and can also exchange messages with each other. The stars represent the gateways, while the squares stand for the bridges. The dashed line in communication topology is the LoRa connection, while the complete line and weight in the bridge network are the road and distance between bridges. As a prerequisite, the LoRa module of end-devices is designed to be at least Class B during an emergency, making them reachable at preconfigure times. Gateways have eight channels (sub-bands), allowing sufficient capability for uplink and downlink, thereby minimizing the duty-cycle influence, and can transmit messages through LoRa between each other.

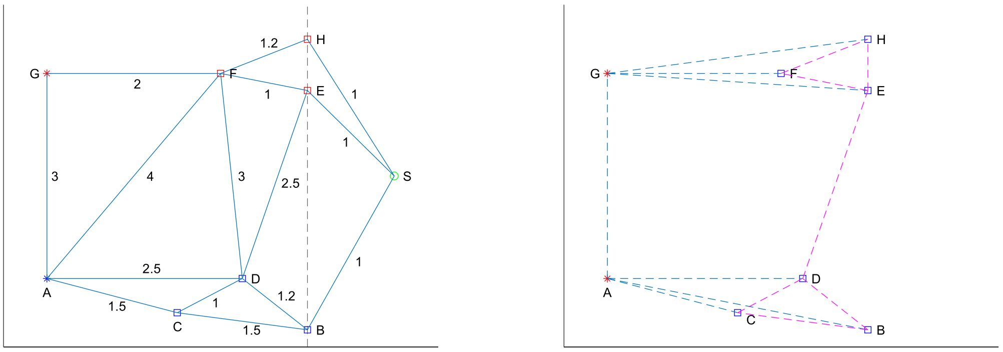


Figure 33 (1) bridge network; (2) communication topology

For simplicity, there are only two conditions for bridge serviceability in the simulation, i.e., Y- available and N - closed. The route planning is only updated when a gateway or sensor node receives the message that a bridge becomes closed, and the affected weights become infinitely great. The information is encoded as a message of characters indicating the bridge's location and condition. For example, “BN” means bridge B becomes unavailable. Because LoRa gateways are usually built on the

Raspberry Pi, they have sufficient computing capability to find the new shortest path via the Floyd algorithm (considering all the nodes) with a computational complexity of $O(n^3)$ and space complexity of $O(n)$, where n is the number of nodes. Similarly, the sensor nodes built on Arduinos can also perform computing to find the new shortest path from their own to the safe area via the Dijkstra algorithm with computational complexity $O((n+m)logn)$ and space complexity $n+m$, where n is the number of nodes and m is the number of edges in the graph. Therefore, if cloud servers become unresponsive temporarily, the dynamic route planning for evacuation can still work as the following procedures.

```
Procedures: dynamic route planning for evacuation
  Input: information of bridge locations and conditions "$BN$", "$CN$", ... "$HN$"
  Output: shortest evacuation path for each node $path(n_i)$
while update messages from unavailable bridges do
    if cloud servers become unresponsive then
        transmit messages through LoRa between gateways;
        compute $p(n_i)$ on gateways via Floyd algorithm;
        if gateways become unresponsive then
            relay messages through LoRa via sensor nodes;
            compute $p(n_i)$ at each sensor node via Dijkstra algorithm;
return $path(n_i)$;
```

The simulation is initialized with all the bridges available. When bridge B becomes closed in the simulation, and cloud servers are out of the connection, the gateways can transmit the messages through LoRa and perform dynamic route planning. Moreover, when the gateways become unresponsive, the sensor nodes will relay the message through LoRa to each other and find the shortest evacuation route, which becomes a decentralized mode. The difference between gateway-based and sensor-node-based route planning is shown in Figure 34.

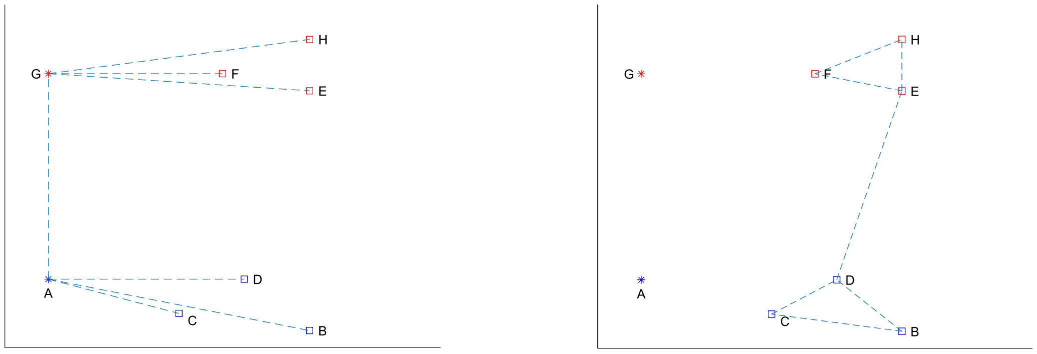


Figure 34 (1) gateway-based route planning (2) sensor-node-based route planning

The results are shown in Table 6. Take “BN” as an example. When the cloud server becomes unresponsive, it will result in new route planning at nodes A, B, C, and D based on the gateways. The downlink instruction message can be encoded as “BDES CDES DES” (i.e., 13 bytes), which will take up to 2.8017 seconds of airtime at the mode SF12/125kHz in both uplink and downlink. Moreover, when gateways become unresponsive, “BN” can be relayed through LoRa to all the nodes (i.e., $B \rightarrow C, D \rightarrow E \rightarrow F, H$). The communication time cost is up to 3.4653s. This experiment demonstrates the excellent fault-tolerant capability of the proposed framework for DT services to endure a temporary loss of communication, especially under emergent situations. In practice, PE (i.e., multiple bridges) in the proposed framework becomes a resilient and self-adaptive subsystem under such conditions, of which the behaviour can be predicted and simulated in the cloud, so PE and VE can be re-synchronized seamlessly when the communication recovers.

Table 6 Simulation result for dynamic route planning

Nodes	Route and Distance	Route and Distance	Communication Time
A	Initial / A C B S and 4	BN / A D E S and 6	2.8017s / 3.4653s
B	Initial / B S and 1	BN / B D E S and 4.7	2.8017s / 3.4653s
C	Initial / C B S and 2.5	BN / C D E S and 4.5	2.8017s / 3.4653s

D	Initial / D B S and 2.2	BN / D E S and 3.5	2.8017s / 3.4653s
E	Initial / E S and 1	BN / E S and 1	0/ 3.4653s
F	Initial / F E S and 2	BN / F E S and 2	0/ 3.4653s
G	Initial / G F E S and 4	BN / G F E S and 4	0/ 3.4653s
H	Initial / H S and 1	BN / H S and 1	0/ 3.4653s

4.4.4 Comparative analysis

A comparative analysis between the proposed framework and the current bridge DTs is presented in Table 7. As can be seen, the proposed AIoT-informed DT framework and the developed cross-platform prototype can handle massive heterogeneous data efficiently using AI-based edge computing and perform DT services in near real-time, even under the communication-constraint circumstances, i.e., LPWAN. Moreover, it has excellent fault tolerance, which can endure a temporary loss of communication rather than failing and is scalable to support single or multiple bridges in a large area.

Table 7 Compare the proposed DT framework and existing bridge DTs

Features	**Proposed Framework**	**cDTSHM** [38]	**Broo et al.** [25]	**Shim et at.** [21]	**Jeong et al.** [228]
Level	Prototype	Prototype	Pilot project	Concept	Prototype
Data type	Heterogenous	time-series	time-series	Heterogenous	time-series
Data collection	Automatic	Semi-automatic	Automatic	Manual	Semi-automatic
Pre-processing	Edge	Fog layer	Cloud	Cloud	Cloud
Computing	Edge & Cloud	Cloud	Cloud	Local server	Cloud
Communication	LPWAN	5G	Ethernet and 4G	N/A	4G
Resilience	Yes	No	No	N/A	No
HMI	Web	Web	Web	Desktop	Web

Time delay	Near real-time	Near real-time	Near real-time	Periodic	Near real-time

4.5 Summary

With the development of sensing and IoT technologies, massive heterogeneous data from regular inspection and real-time monitoring has become a challenge for bridge DT synchronization. However, when DT implementation concerns bridge locations, it may have many issues, such as restricted communication. Meanwhile, most existing bridge DTs are cloud-based and rely on excellent communication without consideration of system resilience to endure a temporary loss of communication. This work proposed an AIoT-informed DT communication framework to support bridge O&M in a communication-constraint environment with high efficiency, low latency, and excellent fault tolerance.

Firstly, the research indicates that the time delay of DT services consists of computation and communication time costs, which depend on computational and communication complexity, respectively, and reveals the distinct impact of their sequence on time consumption for DT services, i.e., edge computing can help to reduce time delay significantly when communication time is dominant in the process. Information hierarchy (i.e., DIKW) is leveraged to indicate how to reduce communication complexity using AI-based edge computing theoretically.

Moreover, two-way communication between edge and cloud is recommended to satisfy the restricted communication with minimal complexity for big-data analysis, which involves different data sources owned by edge and cloud, thereby decreasing the time delay. AI-based edge computing can enable the system with resilience to endure a temporary loss of communication, such as preliminary analysis and decision-making, which is especially beneficial to the safety of physical bridges and public users when a disaster is happening or is predicted to happen. Furthermore, a hierarchical communication architecture with excellent fault tolerance can be designed based on LPWAN and the mesh network for different-level tasks.

Then, a bridge DT system is idealized mathematically, including state-space representation with time delay and inequalities for hardware processing capability. Meanwhile, the data flow for DT services and the resilience of the proposed framework are demonstrated based on Petri-net modelling with token and conditional probability. Furthermore, the framework is developed to the level of a prototype with cross-platform integration for bridge O&M, including AI-based edge computing, LPWAN communication, cloud servers, MQTT protocols, and a web-based platform with both GIS and BIM.

Finally, the proposed framework and prototype are validated with different cases for bridge O&M, including drone-enabled inspection, VBM, and dynamic evacuation. The DL model is trained on Google CodeLab and then converted to a tinyML version for deployment to enhance the efficiency of AI-based inference at the edge. The results demonstrate that 1) the proposed framework can achieve DT synchronization during drone inspection in near real-time under communication-constraint circumstances such as LPWAN; 2) the prototype can achieve similar performance to the previous cloud-based DT [26] in near real-time for vibration-based SHM without relying on excellent communication and has extra resilience; 3) the framework can achieve excellent fault tolerance for DT services through the hierarchical communication architecture to endure a temporary loss of communication at different levels for single or multiple bridges in a large area. These benefits can contribute directly to the efficiency and safety of bridge O&M through DT.

The proposed DT framework and prototype will be implemented on a real-world bridge in the UK for practical application in the next step. This framework is also promising for federated learning to protect privacy because different stakeholders prefer to preserve their AI models derived from specific domain knowledge and experience rather than share them on the cloud. Although this framework has many benefits, which can contribute directly to the efficiency and safety of bridge O&M through DT, it still has some limitations. For example, though LPWAN can reduce the power consumption of communication significantly, AI-based edge computing raises a high requirement for power supply according to the tasks and algorithms, which can be an issue under resource-constraint circumstances. Hence, edge-based AI can only perform preliminary analysis and decision-making currently. Therefore, the sustainable power supply for edge devices and the trade-off between edge and cloud in data storage and computation have great research significance in the future.

Chapter 5 Few-shot bridge image damage detection

Autonomous bridge visual inspection is a real-world challenge due to various materials, surface coatings, and changing light and weather conditions. Traditional supervised learning relies on many annotated data to establish a robust model, which requires a time-consuming data acquisition process. This chapter proposes a few-shot learning (FSL) approach based on improved ProtoNet for damage detection with just a few labelled examples. Feature embedding is achieved through cross-domain transfer learning from ImageNet instead of episodic training. The ProtoNet is improved with embedding normalization to enhance transduction performance based on Euclidean distance and a linear classifier for classification. The approach is explored on a public dataset through different ablation experiments and achieves over 94% mean accuracy for 2-way 5-shot classification via the pre-trained GoogleNet after fine-tuning. Moreover, the proposed fine-tuning methods based on a fully connected layer (FCN) and Hadamard product are demonstrated with better performance than the previous method. Finally, the approach is validated using real bridge inspection images, demonstrating its capability of fast implementation for practical damage inspection with weakly supervised information.

5.1 Theoretical foundation

5.1.1 Few-shot problem definition

ML is said to learn from experience E to some classes of task T, and the performance is measured by P [229], e.g., E – ImageNet dataset, T – object recognition, and P – classification accuracy. Few-shot learning is a specific type of ML problem where E contains only a little supervised information for the task T. In the few-shot setting, the dataset D is separated into $D_{sup\,port}$ and D_{query}, as shown in Eq. 15 and Eq. 16. I is a very small integer, commonly from 1 to 5. In a standard N-way K-shot classification task, $D_{support}$ comes from N categories (N-way) with K samples (K-shot) per category, so there are I=$N \times K$ support examples. D_{query} contains samples from the same N categories with Q samples per category. The goal is to classify the $N \times Q$ images into N categories based on the limited supervised information from $D_{support}$ [230].

$$D_{support} = \{(x_i, y_i)\}_{i=1}^{I=N \times K} \tag{15}$$

$$D_{query} = \{x_j\}_{j=1}^{J=N \times Q} \tag{16}$$

Where N is the number of categories; K is the number of samples (i.e., the support items); x_i is the support item; y_i is the corresponding category for the support item; x_j is the query item.

Let $p(x, y)$ as the joint probability distribution of input x and label y. h is the hypothesis model mapping from x to y. Few-shot classification aims to learn h from $D_{support}$ for prediction and then test it in D_{query}. Here, h is parameterized as $h(\theta)$. The algorithm aims to find the optimal θ for $D_{support}$ in the vector space H. The model h performance is evaluated through the loss function $L(\hat{y}, y)$ between the prediction value $\hat{y} = h(x; \theta)$ and the actual value y.

Assuming vector space H, task T, and distribution $p(x, y)$, to minimize the loss function $L(\hat{y}, y)$ equals to minimize the expected risk $R(h)$ with appropriate θ, which can be indicated in Eq. 17.

$$min\, R(h) = \min \int L(h(x; \theta), y) dp(x, y) = \min\ \mathbb{E}[L(h(x; \theta), y)] \tag{17}$$

In practice, posterior distribution from data sampling is utilized to approach $p(x, y)$ through ML. However, as $p(x, y)$ is unknown, the empirical risk $R_I(h)$ is used to estimate $R(h)$, as indicated in Eq. 18.

$$R(h) \approx R_I(h) = \frac{1}{n} \sum L(h(x_i; \theta), y_i) \tag{18}$$

Hence, there will be three different optimal solutions [231], which are: 1) $\hat{h} = argmin\, R\,(h)$ – global optimal solution; 2) $h^* = argmin_{h \in \mathrm{H}}\, R\,(h)$ – optimal solution in hypothesis space H; 3) $h_I = argmin_{h \in \mathrm{H}} R_I(h)$ – optimal solution in H for $R_I(h)$. Moreover, with model h trained from a random set for a task, its total error consists of two parts: 1) approximation error $\varepsilon_{app}(H)$ caused by the difference between the hypothesis space H and the global space; 2) estimation error $\varepsilon_{est}(H, I)$ is the impact of using empirical risk $R_I(h)$ instead of expected risk $R(h)$. Here, I is the set of training data. In theory, as the training set increases, $\varepsilon_{est}(H, I)$ converges to zero, as shown in Eq. 19.

$$\lim_{I \to \infty} \varepsilon_{est}(H, I) = \lim_{I \to \infty} \mathbb{E}[R(h_I) - R(h^*)] = 0 \tag{19}$$

However, as few-shot learning lacks plenty of training data, it becomes difficult to use $R_I(h)$ approaching $R(h)$ accurately. Therefore, the most difficulty of few-shot learning is the gap between the empirical best $h_I(I)$ and hypothesis best $h^*(H)$.

5.1.2 Meta-learning and feature embedding

Meta-learning approaches aim to learn prior knowledge from a series of training tasks to solve a new task. It includes hallucination-based (learning to augment), initialization-based (learning to fine-tune), and metric-based (learning to compare) approaches. The hallucination-based approaches [122]–[124] aim to generate more training examples of novel classes through data augmentation to alleviate the issue of insufficient data. The initialization-based approaches, e.g., MAML [232], Reptiles [127], and LEO [233], aim to learn the optimal hyperparameter initialization to reach convergence with only a small number of data samples. The metric-based approaches, e.g., MatchingNet [234], ProtoNet [133], and RelationNet [137], aim to project data into an embedding space in which similar objects are close to each other and vice versa. The transductive inference process is to calculate the distance (or similarity) between $x_i \in D_{support}$ and $x_j \in D_{query}$, then the label y_i with the closest distance (or highest similarity) in $D_{support}$ is assigned as y_j in D_{query}. In detail, MatchingNet uses attention calculated from the cosine similarity of extracted features for classification;

ProtoNet uses the mean vector of each class as the cluster centre and Euclidean distance as the metric for classification; RelationNet employs relation module instead of Cosine similarity and Euclidean distance, generating a non-linear classifier based on relation score. These sophisticated meta-learning approaches are usually based on episodic training through a series of related tasks (episodes) sampled from the base dataset to simulate reasoning scenarios [139].

Feature embedding (representation) is used to represent a data point $x_i \in X \subset \mathbb{R}^d$ in a low-dimension space $z_i \in Z \subset \mathbb{R}^m$ $(m < d)$, which is supposed to have three essential assumptions [235], i.e., smoothness, clustering, and manifold. Feature embedding must retain consistent similarities or differences among data points in the original space. Embedding functions are usually in the form of DNN architectures. Note that feature representations through different embedding functions can have different properties, even from the same data point, which can significantly impact the performance of downstream tasks. The hyper-parameters of the embedding function can be learned from prior knowledge or task-specific information, e.g., multiple sophisticated tasks or a related source domain.

The support embedding function and query embedding function are usually the same. The most straightforward way to learn the embedding function is to train a model in the support set through supervised learning. Still, its parameters are prone to overfitting or difficult to converge under few-shot conditions. Hence, many existing few-shot learning works tackle this problem based on meta-learning, i.e., trained on a series of invariant tasks and then generalized to the target task. However, cross-domain transfer learning has been recently demonstrated as an effective way to initialize the feature embedding functions for few-shot classification [136], [138] instead of meta-learning.

5.1.3 Transfer learning and fine-tuning

Transfer learning focuses on storing the knowledge learned while solving one task T_S

in a source domain $\mathbb{R}_S$ and applying it to a different but related task T_T in a target domain $\mathbb{R}_T$. The correlative research problems, such as multi-task learning and domain adaption, are also related to few-shot learning and meta-learning [236]. In multi-task learning, the hypothesis space of each task strongly correlates with each other. This correlation (i.e., prior knowledge) can be represented through sharing hyperparameters of DNNs. According to explicit or implicit constraints in parameter space, the sharing methods can be classified into soft parameter sharing, which does not place a strong constraint on parameters but encourages them to meet some requirements, such as regulation function L_1 or L_2, and hard parameter sharing, such as freezing specific layers in DNN. The frozen layers can be part of the embedding function or just the classifier, which solidifies the prior knowledge learned from the source task T_S. At the same time, the rest of the network will be updated (i.e., fine-tuning) to adapt the target task T_T in the target domain.

Some meta-learning works have been developed to leverage transfer learning by learning the scaling and shifting functions of DNN weights through episodic training for each task, such as meta-transfer learning [237]. Research [8] has recently demonstrated that cross-domain transfer learning can achieve the comparable performance of (or even overperform) many state-of-the-art meta-learning approaches in few-shot classification. Moreover, fine-tuning can enhance average accuracy by 1%-2% on the CUB and ImageNet datasets [136], [138]. This progress enables few-shot classification to be exempt from episodic training and become "training-free" like [139] by using pre-trained DNN backbones from a large-scale public dataset (e.g., ImageNet) for feature embedding.

5.2 Few-shot damage detection approach

5.2.1 Proposed architecture

The proposed approach for bridge damage detection is derived from the ProtoNet [133], which consists of episodic training through a series of related tasks and prototypical transduction based on Euclidean distance for few-shot classification. Its improvement includes three aspects: (1) previous episodic training is replaced with cross-domain transfer learning from ImageNet for "training-free" feature embedding; (2) embedding normalization is integrated to reduce domain variation and enhance the original ProtoNet performance based on Euclidean distance; (3) the fine-tuning methods based on fully connected network (FCN) and the Hadamard product can achieve better performance in fewer epochs compared to the previous transductive fine-tuning [138].

The approach architecture is shown in Figure 35 with an example of 2-way 3-shot crack detection, and the steps are shown below.

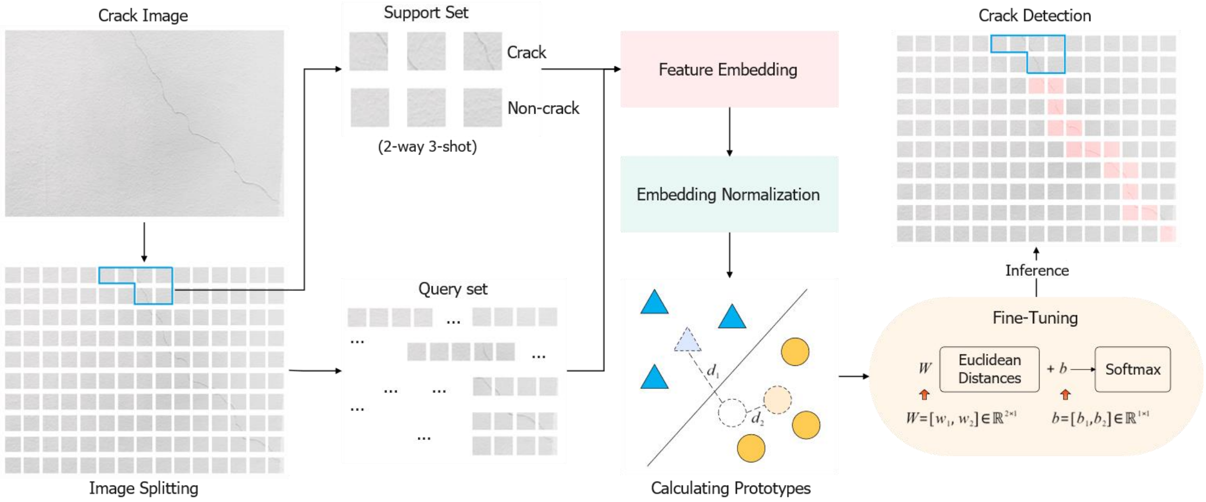


Figure 35 Proposed approach for few-shot damage detection

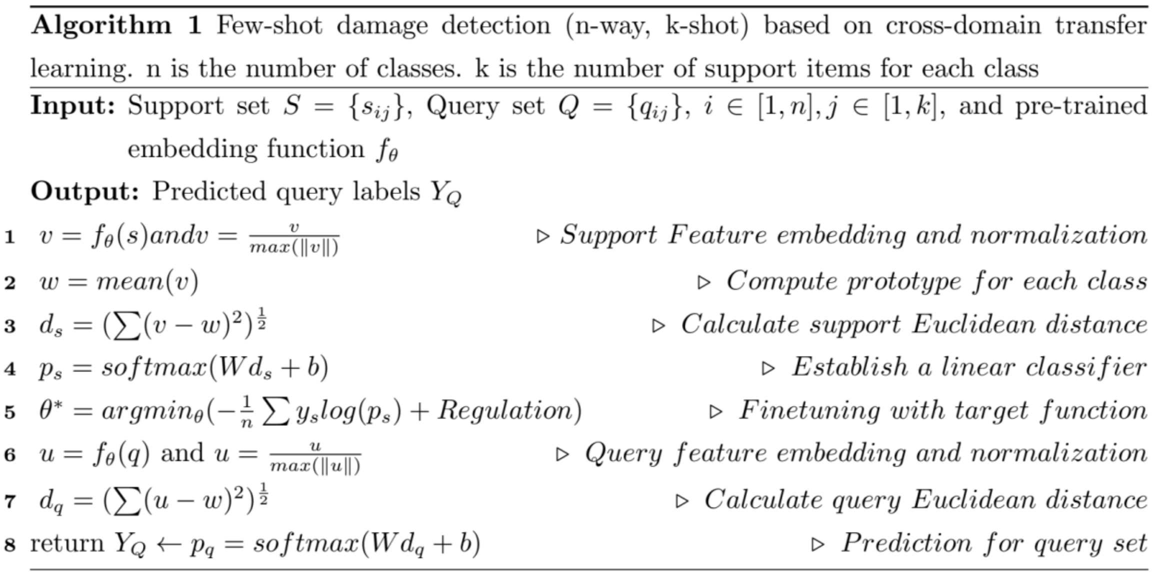
1) **Image splitting into support and query sets** – an inspection image is split into multiple patches, in which the support and the query items are selected, respectively. Here, the patches marked with the blue boundary are picked up as the support set, while the remaining patches are taken as the query set.

2) **Feature embedding** (cross-domain transfer learning) – the pre-trained DNN backbones from ImageNet are applied on both support and query items for feature embedding, which not only enables the feature embedding to be exempt

from episodic training but also makes the process become "training-free" (no need to be trained from scratch).

3) **Feature normalization** – normalization is employed after feature embedding to reduce domain variation.
4) **Calculating prototypes** – the mean vector of the support feature embeddings is calculated as the prototype for each class, and the initial transductive inference can be taken based on Euclidean distance.
5) **Fine-tuning** – fine-tuning is employed to improve the linear classifier further using the support examples and the derived prototypes.
6) **Inference** – finally, the damage type, location, and skeleton can be obtained based on the inference for each patch. Meanwhile, the obtained prototypes and fine-tuned classifier can be applied to a new image to detect the specific defect.

The pseudocode of the algorithm is shown below:



```
Algorithm 1 Few-shot damage detection (n-way, k-shot) based on cross-domain transfer learning. n is the number of classes. k is the number of support items for each class
Input: Support set $S = \{s_{ij}\}$, Query set $Q = \{q_{ij}\}$, $i \in [1, n], j \in [1, k]$, and pre-trained
        embedding function $f_\theta$
Output: Predicted query labels $Y_Q$
$v = f_\theta(s) and v = \frac{v}{max(\|v\|)}$    ▷ Support Feature embedding and normalization
$w = mean(v)$    ▷ Compute prototype for each class
$d_s = (\sum(v - w)^2)^{\frac{1}{2}}$    ▷ Calculate support Euclidean distance
$p_s = softmax(W d_s + b)$    ▷ Establish a linear classifier
$\theta^* = argmin_\theta(-\frac{1}{n}\sum y_s log(p_s) + Regulation)$    ▷ Finetuning with target function
$u = f_\theta(q)$ and $u = \frac{u}{max(\|u\|)}$    ▷ Query feature embedding and normalization
$d_q = (\sum(u - w)^2)^{\frac{1}{2}}$    ▷ Calculate query Euclidean distance
return $Y_Q \leftarrow p_q = softmax(W d_q + b)$    ▷ Prediction for query set
```

5.2.2 Domain adaption and transduction

In principle, the pre-trained DNN backbones and weights based on prior knowledge (e.g., from the related source domain) can help to constrain the hypothesis space into a smaller one for few-shot classification, as shown in Figure 36, thereby achieving less ε_{est} quickly and better $R_I(h)$. The left ellipse shows the normal ε_{est} based on a large dataset, which is the goal to pursue. The middle one shows a bigger ε_{est} based on a small dataset (i.e., under FSL conditions), while the right one shows a decreased ε_{est} in a constrained hypothesis space by prior knowledge.

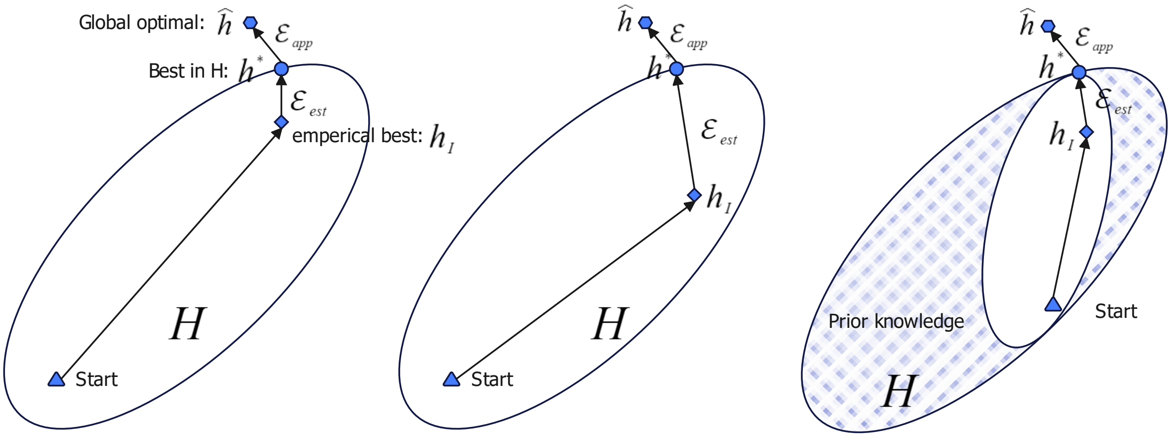


Figure 36 Decreased ε_{est} in constrained hypothesis space by prior knowledge [231]

In the embedding module, the pre-trained DNN backbone (feature extractor) learned from 1000-class ImageNet of 12 million images is employed as the embedding function $f_\theta(x_i)$ for both support and query sets. Note that the object classes of ImageNet do not include the specific defects for detection, i.e., the source domain has a vast difference from the target domain. The embedding function $f_\theta(x_i)$ can be derived from supervised or self-supervised learning, as shown in Figure 37. The former includes different DCNNs and vision transformers. The latter mainly involves masked image modelling (MIM) approaches, such as masked autoencoder (MAE) [238] or BEiT [239].

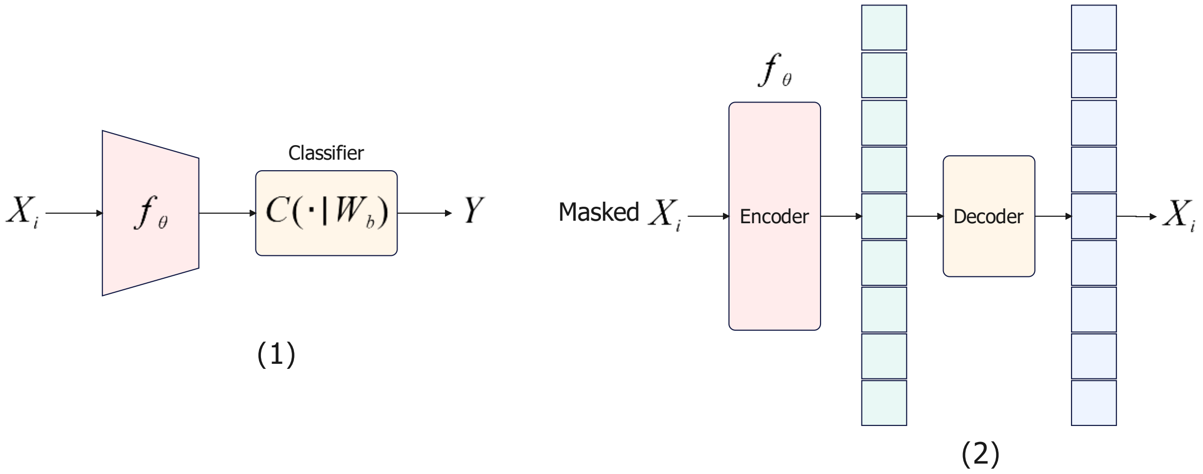


Figure 37 (1) $f_\theta(x_i)$ from supervised learning; (2) $f_\theta(x_i)$ from self-supervised learning

Although the hardcoded mean and the standard deviation obtained statistically from ImageNet, i.e., $\mu = [0.485,0.456,0.406]$ and $\sigma = [0.229,0.224,0.225]$ can be employed for image transformation, it cannot guarantee the normalization in the target domain. Hence, normalization according to Eq. 20 (v is the embedding vector) is required for the obtained feature embeddings to minimize domain variation.

$$v_{norm} = \frac{v}{max(\|v\|_2)} \tag{20}$$

In the transductive inference, the mean vector of the support embeddings is computed as the prototype for each class. Then, the distances from the query embedding to each prototype are calculated. Consequently, the query item can be predicted as the closest prototype. The metrics commonly include Euclidean distance and cosine similarity, as indicated in Eq. 21 and Eq. 22. Here, v and w are the query and prototype embedding vectors, respectively. As seen, embedding normalization enables the transduction based on Euclidean distance and cosine similarity to start from the same circumstance.

$$d = dist(v,w) = \left(\sum |v-w|^2\right)^{\frac{1}{2}} \tag{21}$$

$$s = cos\,\theta = \frac{v^T w}{\|v\|_2 \cdot \|w\|_2} \tag{22}$$

5.2.3 Loss function and fine-tuning

In the proposed architecture, the linear classifier $W^T x + b$ is utilized for few-shot classification. x can be either the query embedding vector v or the distances d between the query item and the prototypes. The softmax function is utilized as the output layer to convert the result to a probability distribution $p_i \in [0,1]$ for each class, as shown in Eq. 23.

$$p_i = soft\,max(x_i) = \frac{e^{x_i}}{\sum_{n=1}^{N} e^{x_n}} \tag{23}$$

Then, the loss function L is defined based on binary cross-entropy, as indicated in Eq. 24.

$$L = \frac{1}{N}\sum_i L_i = -\frac{1}{N}\sum_i [y_i\, log_2(p_i) + (1 - y_i)\, log_2(1 - p_i)] \tag{24}$$

Where y_i is the example label (0 or 1); p_i is the probability of y_i for the example i.

As the support set is relatively small under few-shot conditions, the Shannon Entropy (Eq. 25) is introduced as the regularization item to alleviate overfitting due to increased uncertainty in classification, as shown in Figure 38. It is similar to the transductive fine-tuning method in [138], but the entropy $H(x)$ is calculated based on the support set rather than the query set because the model aims to be trained and fine-tuned before seeing all the query items in the practical inspection.

$$H(x) = -\sum_i p_i \cdot log_2 \frac{1}{p_i} \tag{25}$$

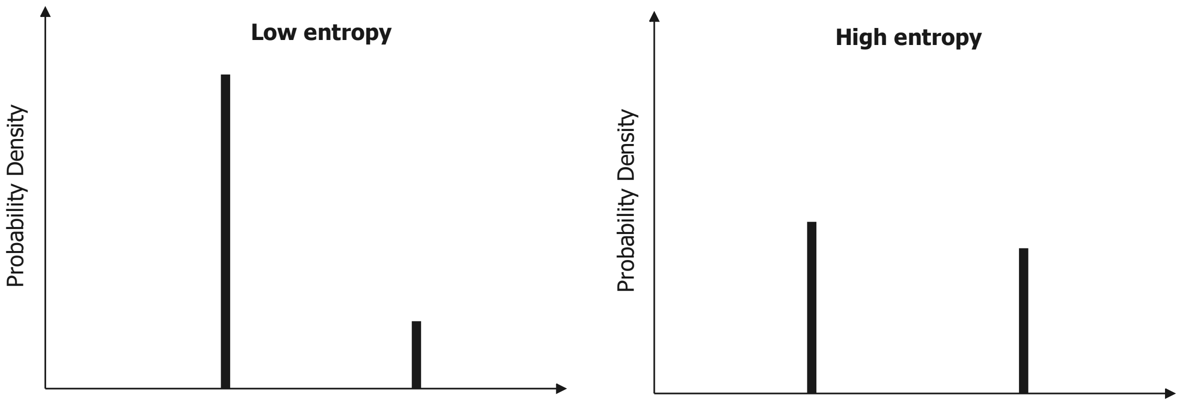


Figure 38 Entropy increases along with uncertainty rising in binary classification

Hence, the fine-tuning step solves Θ^* to minimize the target function indicated in Eq. 26.

$$\Theta^* = \underset{\Theta}{argmin}(-\frac{1}{N}\sum_i [y_i \, log_2(p_i) + (1-y_i) \, log_2(1-p_i)] - \frac{1}{N}\sum_i p_i \, log_2(p_i)) \quad (26)$$

5.3 Experiments and approach validation

5.3.1 Experiment preparation

An image dataset created for automatic bridge crack detection in [107] is employed for ablation studies using the proposed architecture for few-shot crack classification. The images were collected from real concrete bridges, including the 2014 background and 4055 crack images (224×224). The dedicated CNN in the previous research [107] can reach 96.37% accuracy on the test set (train-test split of 80%:20%) based on supervised learning. Here, the experiment aims to explore the performance of the proposed approach for few-shot crack classification (2-way 1-shot or 2-way 5-shot) on the test set, i.e., with no access to the training set for supervised learning. It can mimic the situation for crack identification without a pre-trained supervised model. The query accuracy is illustrated in a boxplot based on 5000 samplings, recommended to compare FSL performance by [138]. The random state remains unchanged to guarantee the reliability of ablation experiments. The experiments are taken on Google

CodeLabs. The code is generated based on the original ProtoNet from a public GitHub project (https://github.com/sicara/easy-few-shot-learning).

5.3.2 Ablation studies

5.3.2.1 Domain adaption and normalization

The experiment starts with 2-way 1-shot and 2-way 5-shot crack identification. The ResNet18 backbone, popular in previous few-shot learning research [138], [240], [241], is employed as the feature embedding function. Its parameters are pre-trained on ImageNet, and the hardcoded mean $\mu = [0.485, 0.456, 0.406]$ and standard deviation $\sigma = [0.229, 0.224, 0.225]$, derived from ImageNet statistically, are utilized for image transformation. The raw and hardcoded-transformed images can be shown in Figure 39. The image size is 224×224.

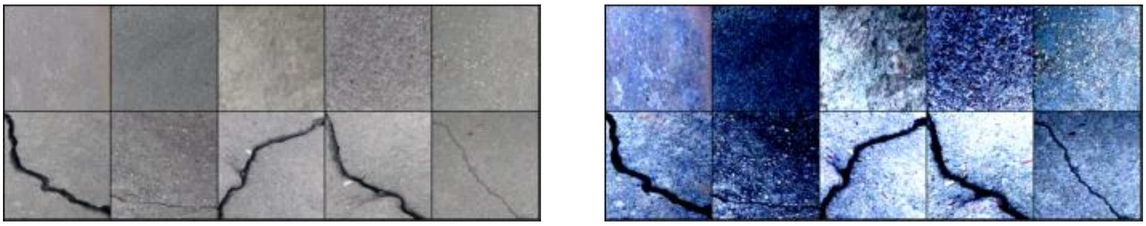


Figure 39 Raw images (left) and hardcoded-transformed images (right)

The performance of architecture with and without embedding normalization is explored in the experiment. Moreover, Euclidean distance and cosine similarity are tested as the evaluation metric. The results are shown in Figure 40. Here, the annotation with raw and hard represents raw and hardcoded-transformed images, respectively; Eu indicates that the result is based on Euclidean distance of raw embedding vectors, while Eu_norm stands for Euclidean distances of embedding vectors after normalization; Cosine means using cosine similarity of raw embedding vectors.

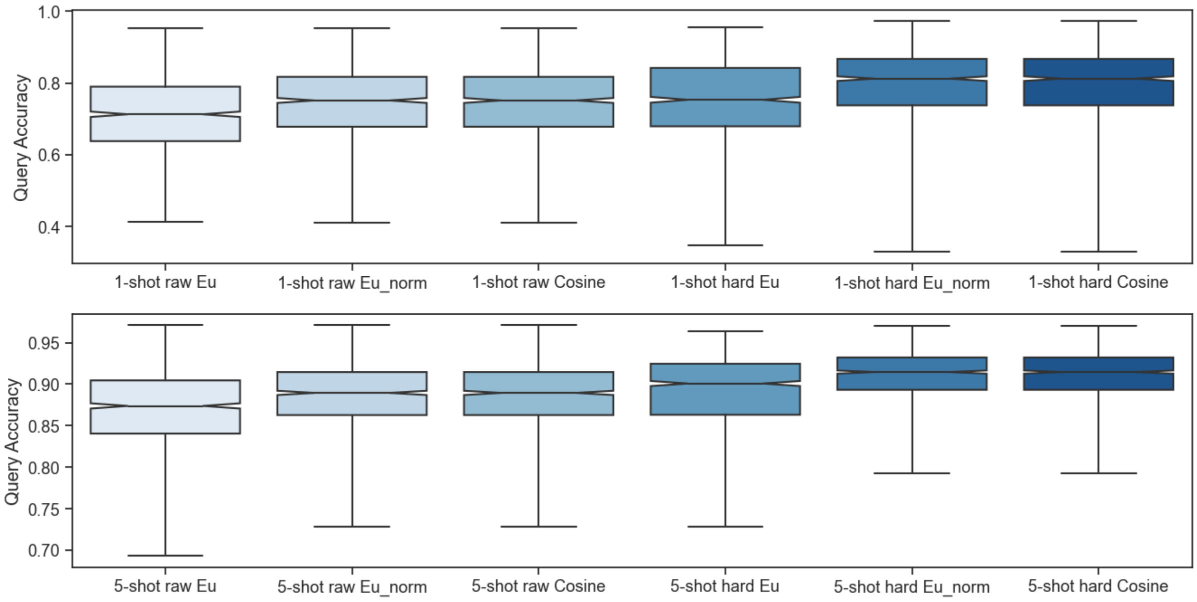


Figure 40 1-shot and 5-shot crack identification with pre-trained ResNet18 (224×224)

As can be seen, hard-coded transformation (i.e., hard) can significantly improve both 1-shot and 5-shot performance. After hard-coded transformation, it is shown with higher mean accuracy and narrower value distribution, i.e., interquartile range (IQR). IQR is calculated as $IQR = Q_3 - Q_1$ (Q_1 – the first quartile; Q_3 – the third quartile). Moreover, the Euclidean distance of normalized embeddings (i.e., Eu_norm) performs much better than the raw Euclidean distance (i.e., Eu_raw). The former has the equivalent performance as the cosine similarity, demonstrating that embedding normalization can bridge the gap between Euclidean distance and cosine similarity in the metric-based transduction for the few-shot classification in this dataset. Furthermore, 5-shot performs much better than 1-shot in accuracy and IQR, promising to be comparable with the dedicated supervised learning in previous research. Meanwhile, as the experiment aims to validate the proposed approach and figure out the appropriate conditions (such as feature embedding functions and fine-tuning methods) for practical application under weakly supervised scenarios, the 2-way 5-shot classification is adopted for the following experiment.

5.3.2.2 *Different embedding functions*

A series of ResNet backbones in different depths are employed in the experiment to explore the impact of DNN architecture depths on the few-shot performance. Their parameters are pre-trained on ImageNet. The experiment is conducted for 2-way 5-shot classification, and the approach integrates hard-coded transformation and embedding normalization. Euclidean distance and cosine similarity are tested as the evaluation metric in the experiment. The results are shown in Figure 41.

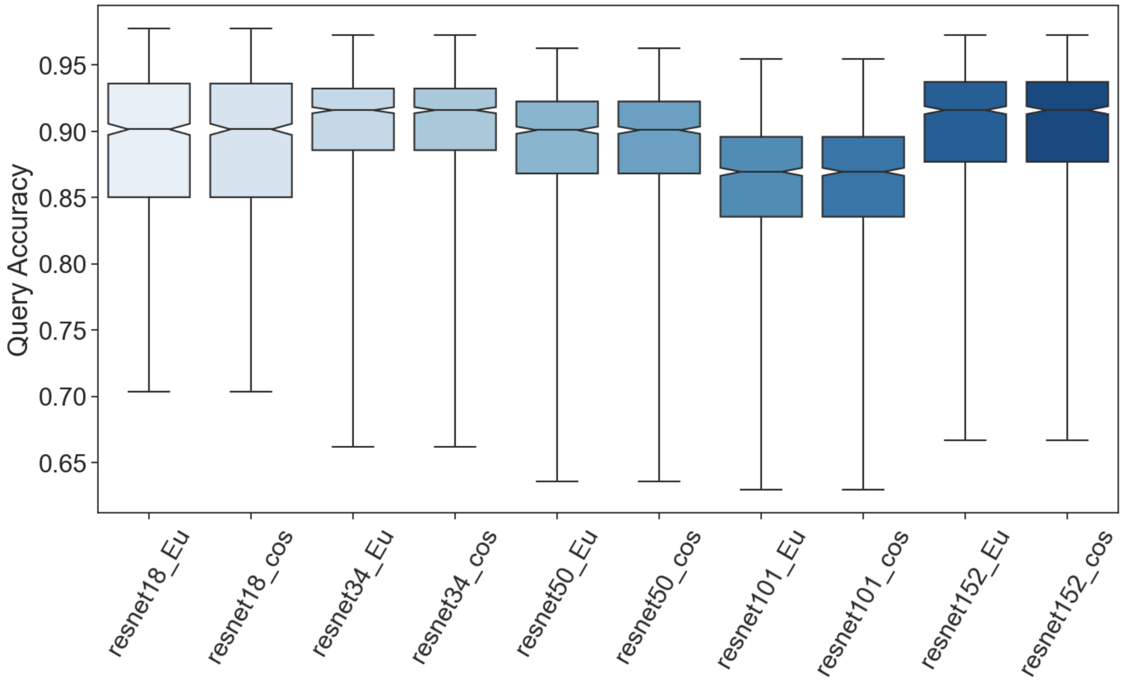


Figure 41 2-way 5-shot performance of ResNet backbones in different depths (84×84)

Here, the images are resized to 84×84 to fit deep ResNets (such as ResNet152) due to CUDA memory limitation, so the ResNet18 performance differs from its previous result in Figure 40 (224×224), i.e., the minimum accuracy drops to nearly 70%. Although the deeper ResNet has higher accuracy for image recognition in ImageNet, the experiment with different pre-trained ResNets for feature embedding cannot see a significant proportional relationship between the performance and the DNN depths for 2-way 5-shot classification, as shown in Figure 41. Therefore, when using the pre-trained DNN backbones as embedding functions, their cross-domain few-shot performance does not necessarily correspond to their original performance in the source domain.

Moreover, different ResNets can perform diversely, even for the same sample. For example, ResNet18 has only 76.5% query accuracy for a sample (i.e., 5-shot for crack and 5-shot for non-crack), while ResNet152 can reach 91% for the same sample. Meanwhile, Euclidean distance and cosine similarity have the equivalent performance as the evaluation metric. Here, the pre-trained backbone ResNet34 has the best performance with the highest mean accuracy of 91.7% and narrower IQR in the series of ResNets for 2-way 5-shot classification in this dataset (images resized to 84×84).

Furthermore, the other prevalent DNN backbones are involved in the experiment, including multiple DCNN architectures and vision transformers (i.e., Swim Transformer and MAE). Their parameters are still pre-trained on ImageNet. The employed DNN models and their embedding dimensions are shown in Table 8.

Table 8 Pre-trained embedding functions and embedding dimensions

Embedding function	**Pre-trained Models**	**Embedding dimensions**	**Input size**
AlexNet	alexnet	9216	84×84
VGG	vgg16	25088	84×84
DenseNet	densenet161	2208	84×84
EfficientNet	efficientnet_v2	1208	84×84
ResNet	resnet34	512	84×84
MobileNet	mobilenet_v3_large	960	84×84
GoogleNet	googlenet	1024	84×84
Swim Transformer	swim_t	768	84×84

MAE	mae_visualize_vit_base	768	224×224

The approach in the experiment is the same as the above for the ResNets, which integrates both hard-coded transformation and embedding normalization, and the experiment is taken under nearly the same conditions. The only difference is that the pre-trained MAE can only be applied on the 224×224 images, which cannot take all the remaining images (except the support images) as the query set due to CUDA limitation. Hence, the experiment with the pre-trained MAE for feature embedding is taken on the original 224×224 images with the randomly selected 50 images per class as the support set every time. In contrast, the experiment with the other pre-trained DNN backbones is taken under the same condition as the above, i.e., with resized images (84×84) and all the left images as the support set. Both Euclidean distance and cosine similarity are tested in the experiment. The results are shown in Figure 42.

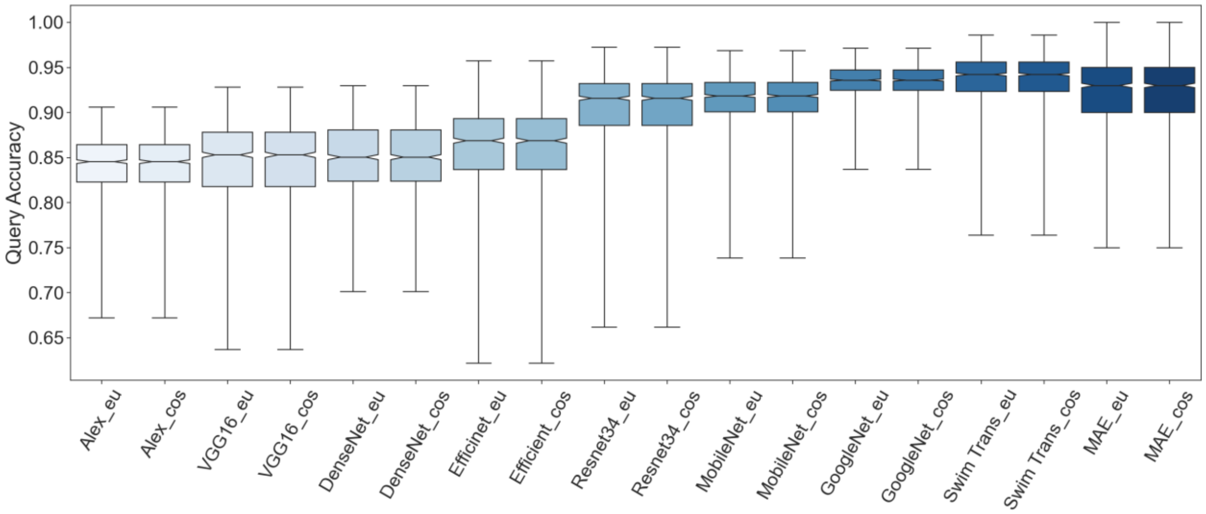


Figure 42 Comparison of different pre-trained DNN embedding functions

As can be seen, the pre-trained DNN backbones can achieve excellent performance for 2-way 5-shot classification. The improved ProtoNet can reach a mean accuracy of over 93% via GoogleNet and Swim Transformer, which proves that ImageNet is a reliable source domain for few-shot crack detection. Note that the pre-trained MAE encoder is derived from self-supervised learning, demonstrating the availability of a training embedding function without supervised information (i.e., labels). It indicates that ImageNet is a reliable source domain for few-shot crack identification based on cross-domain transfer learning. Moreover, the Euclidean distance of the normalized embeddings can achieve the equivalent performance as cosine similarity for the transductive inference.

5.3.2.3 Fine-tuning and comparison

Fine-tuning aims to improve the few-shot classification performance based on transduction after feature embedding through the pre-trained DNN backbones. Its target function can be seen in 3.2.3. Here, three different fine-tuning methods are compared in the experiment, including the Baseline and FCN-based (modified Baseline++) methods, which are inspired by previous research [136], [138], and a proposed method based on Hadamard product (i.e., element-wise product). Meanwhile, fine-tuning with and without the Shannon Entropy regularization (see Eq. 11) is also explored in the experiment. The entropy is calculated based on the support set rather than the query set because the model aims to be trained and fine-tuned before seeing all the query images. This process is different from the previous research [138].

1) The first linear classifier is implemented by adding a linear layer after the normalized feature embedding, similar to the Baseline in [136] and transductive fine-tuning in [138]. Its formula is indicated in Eq. 13, where n is the number of classes ($n = 2$), and m is the embedding dimension. $x_{m\times 1}$ is the normalized feature embedding of each support example. $W_{n\times m}$ is initialized with the prototype matrix $M_{n\times m}$ (i.e., the stack of prototype embedding vectors $[w_1, w_2] \in \mathbb{R}^{1\times m}$) because it can help hyperparameters converge quickly and perform better, as suggested in [138]. $b_{n\times 1}$ is the bias and initialized from 0.

$$y_{n\times 1} = soft\,max(W_{n\times m} \cdot x_{m\times 1} + b_{n\times 1}) \tag{27}$$

2) The second one is adding an FCN after Euclidean distance, as indicated in Eq. 14, which is similar to the Baseline++ in [136] and taken as the modified Baseline++. $d_{n\times 1}$ represents Euclidean distances from a support example to each prototype. $W_{n\times n}$ and $b_{n\times 1}$ are initialized from an identity matrix and 0, respectively.

$$y_{n\times 1} = softmax(W_{n\times n} \cdot d_{n\times 1} + b_{n\times 1}) \tag{28}$$

3) The third one is based on the Hadamard product by adding a linear layer with fewer hyperparameters after Euclidean distance, as indicated in Eq. 15. $d_{n\times 1}$ represents Euclidean distances from a support example to each prototype. $W_{n\times 1}$ and $b_{n\times 1}$ are initialized from 1 and 0, respectively.

$$y_{n\times 1} = softmax(W_{n\times 1} \odot d_{n\times 1} + b_{n\times 1}) \tag{29}$$

Here, the experiment employs the GoogleNet result for fine-tuning as it performs well in query accuracy and IQR. The experiment uses the RMSProp optimizer at the learning rate 0.01 until 2000 epochs. The mean query accuracies and 95% confidence interval of different fine-tuning methods with and without entropy regularization are shown in Figure 43 and Figure 44.

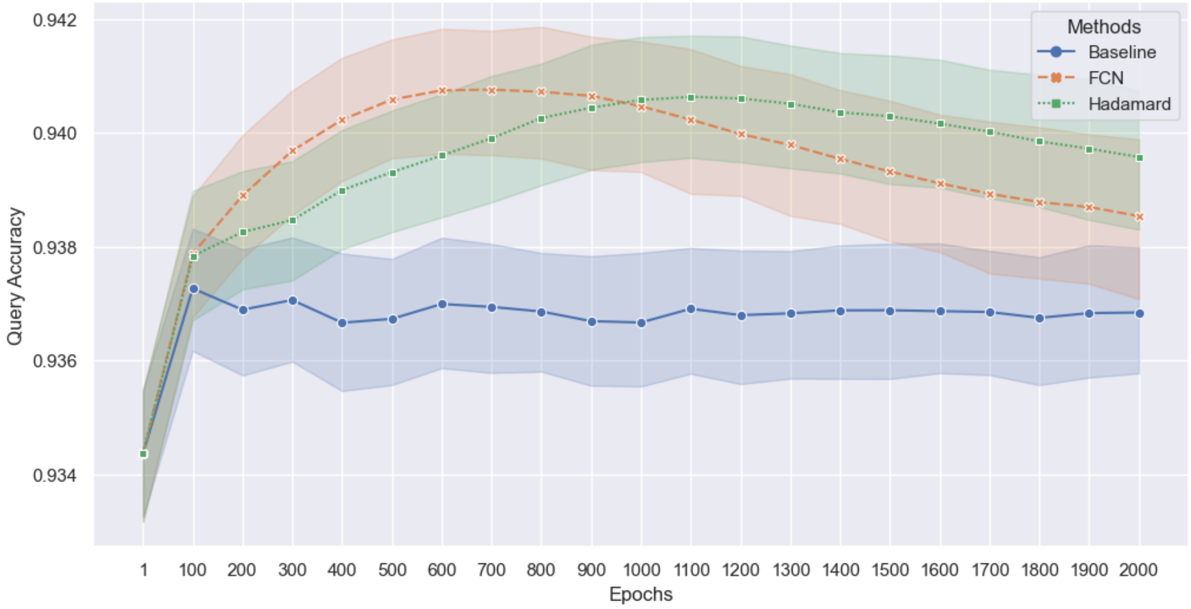


Figure 43 Fine-tuning without entropy regularization

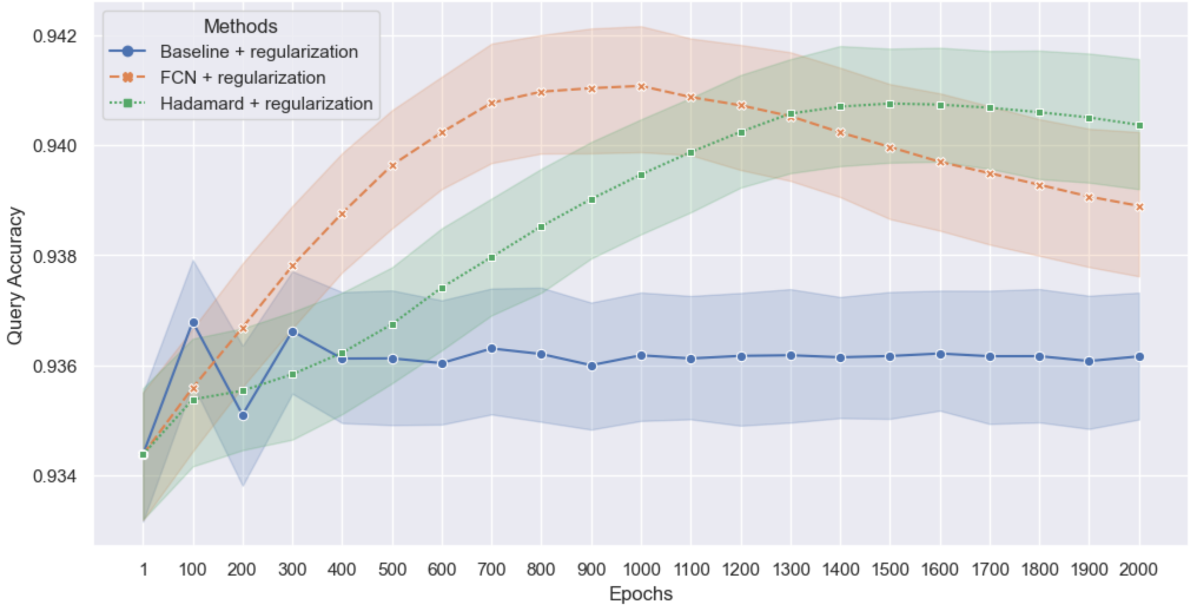


Figure 44 Fine-tuning with entropy regularization

As can be seen, both the FCN-based (i.e., modified Baseline++) and the Hadamard product fine-tuning methods perform much better than the Baseline (i.e., transductive fine-tuning [138]), which can enhance the mean query accuracy from 93.4% to over 94%. Moreover, the FCN-based method can reach the peak faster than the Hadamard-product method in terms of accuracy during fine-tuning. Entropy regularization will slow down the fine-tuning of both methods and postpone their time to reach the peak. After the peak, there is overfitting for both methods. Hence, early stopping should be taken at the epoch number where query accuracy reaches the peak. As can be seen, early stopping can be determined empirically for few-shot crack detection as 600 epochs and 1000 epochs when using the proposed methods without regularization. Similarly, 1000 epochs and 1500 epochs are recommended for both methods with regularization.

In principle, avoiding overfitting in few-shot classification is difficult because the discrepancy between the support examples and the overall items triggers it. If the support examples are representative, fine-tuning by fitting the model to the selected examples can enhance the query accuracy. On the contrary, fine-tuning will deteriorate the model and decrease its generalization capability if the support examples are unrepresentative. It can also be observed that the support set with increased accuracy after the Baseline fine-tuning can get more increment after the FCN and Hadamard-product fine-tuning. At the same time, the other two methods can also amplify the accuracy decrement after the Baseline fine-tuning.

5.3.3 Few-shot damage detection

The approach is also validated with the real bridge inspection images from the CODEBRIM dataset [242]. The images are resized to 1260×840 and split into 150 patches (84×84). A few patches with and without target defects are selected as the support set, while the others are taken as the query set. The embedding function is selected from the pre-trained DNN backbones based on ImageNet, and the classifier is fine-tuned with the support examples. Subsequently, the transductive inference is applied on each query patch using the obtained prototypes and fine-tuned classifier for damage detection. The pre-trained VGG16, VGG19, Swim Transformer, and MAE performed well as embedding functions in the experiment. Here, the results are shown based on the MAE encoder derived from self-supervised learning for feature embedding, in which each patch is resized to 224×224 for inference as required by Vision Transformer (i.e., ViT-Base). Moreover, the time cost is also tested for damage detection using different embedding functions.

An example of 2-way 2-shot crack detection on the real bridge inspection images is shown in Figure 45. The support examples are from the first image in the top right, marked with a blue boundary, and the approach can recognize the crack skeleton with only two shots. The obtained prototypes and fine-tuned classifier can be applied on a new image directly for crack detection in the bottom right. As can be seen, most crack areas can be identified correctly, but a few crack patches were not recognized due to stains, which is related to the approach's robustness.

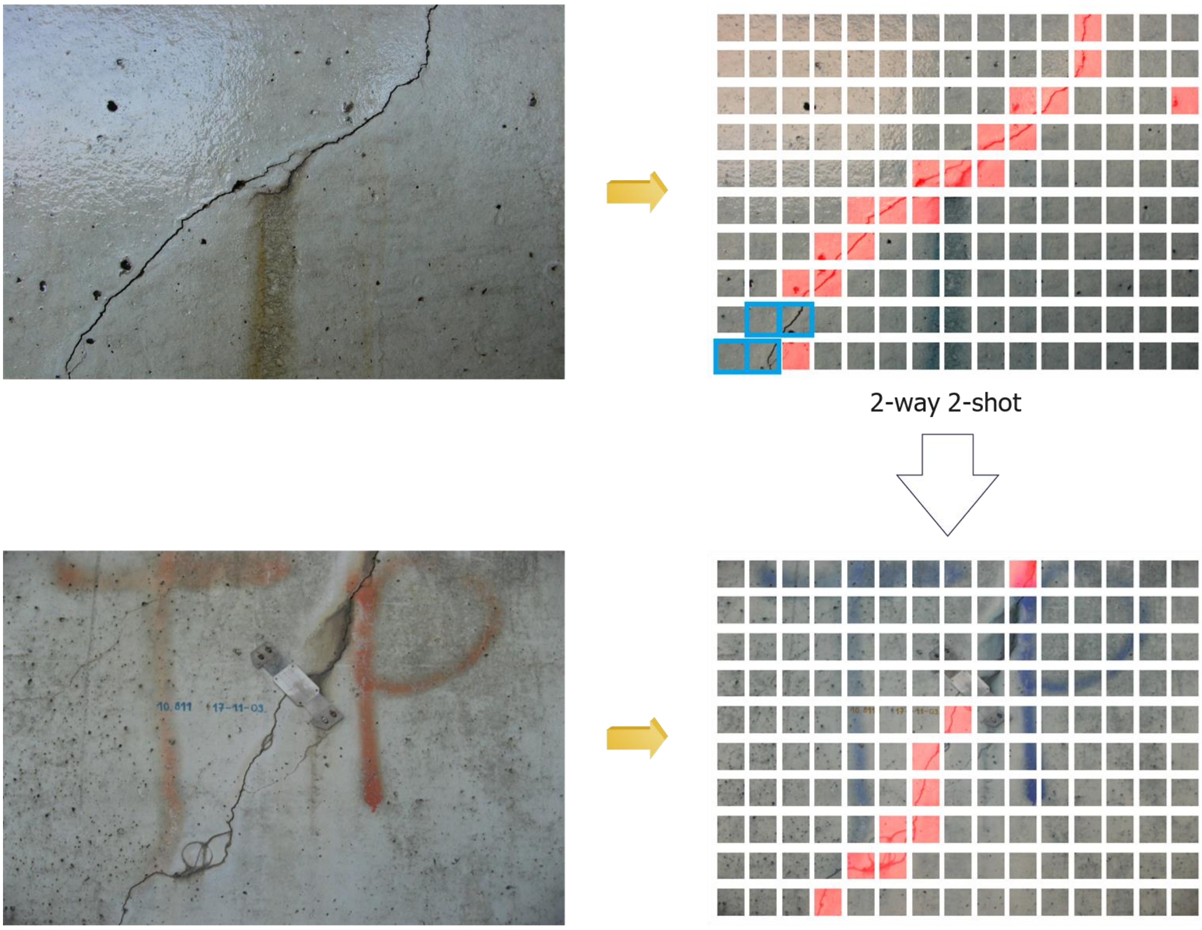


Figure 45 Few-shot crack detection through the approach based on MAE

Spalling with rebar corrosion is another typical defect on the reinforced concrete bridge. An example of 2-way 5-shot spalling detection on the real bridge inspection images is shown in Figure 46. The support patches are from the first image in the top right, marked with a blue boundary, and the approach can recognize the most spalling areas. Similarly, the spalling areas can be identified when applied to a new image in the bottom right through the identical prototypes and fine-tuned classifier.

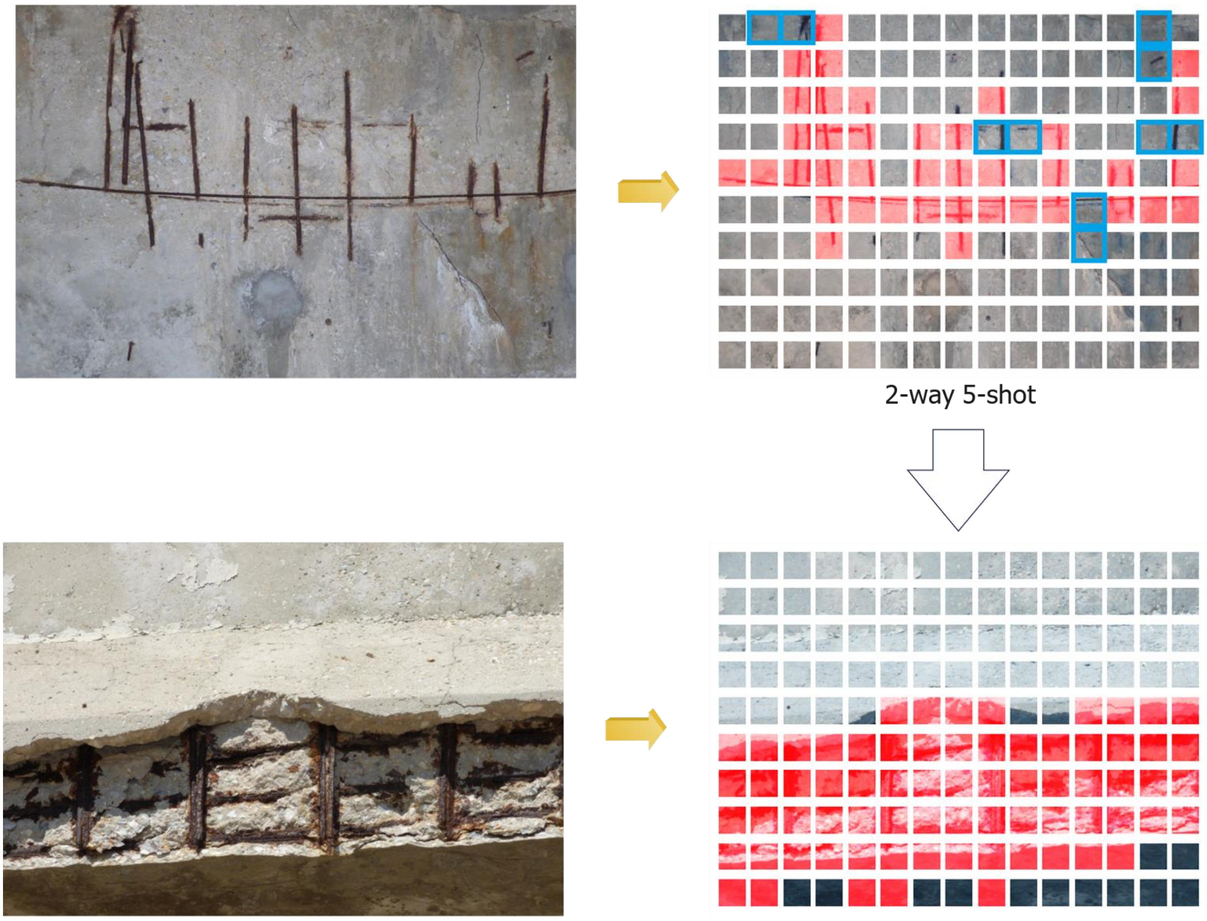


Figure 46 Few-shot spalling detection through the approach based on MAE

The time cost of the approach by using different embedding functions for each patch (84×84) is shown in Table 9. As seen, the time cost increases as the model complexity and input image size increase.

Table 9 Time costs of the approach using different embedding functions

Embedding function	**Pre-trained Models**	**Patch size**	**Time cost**
VGG16	vgg16	84×84	0.08s / patch
VGG19	vgg19	84×84	0.08s / patch
Swim Transformer	swim_t	84×84	0.101s / patch
MAE	mae_visualize_vit_base	224×224 (resized)	0.25s / patch

6.1.1 Summary

The current image-based approaches for drone-enabled bridge inspection still mainly rely on supervised learning, which requires time-consuming data acquisition and labour-intensive data annotation. These inductive approaches are inappropriate for practical damage detection under complex circumstances without enough supervised information, such as different materials, novel defects, and changing light. To solve this issue, this work proposes an approach based on improved ProtoNet for bridge damage detection under few-shot conditions (with only a few annotated examples).

In the approach, feature embedding is achieved by cross-domain transfer learning from ImageNet, which enables the embedding function to be not only exempt from episodic training but also become "training-free", i.e., no need to be trained from scratch. Moreover, after feature embedding, normalization is integrated into the ProtoNet to reduce the domain variation and enhance the transduction performance based on Euclidean distance. The linear classifier is added at the end of the ProtoNet for classification, and fine-tuning based on the support set can be further leveraged to improve the performance.

The approach is explored in a public automatic bridge crack detection dataset through extensive ablation studies. The experiment proves that ImageNet is a reliable source domain for few-shot damage detection and can achieve a mean accuracy of over 94% for 2-way 5-shot classification in the test set via the pre-trained GoogleNet after fine-tuning. The performance is already close to supervised learning using a dedicated CNN architecture. Moreover, the proposed fine-tuning methods based on the FCN and the Hadamard product demonstrated better performance than those in previous research [136], [138]. The time for early stopping can be determined empirically in the experiment. Furthermore, the approach is also validated using real bridge inspection images, demonstrating its capability of fast implementation for damage detection with weakly supervised information and the potential for practical application in near real-time.

Although the approach has the above advantages, it still has a few limitations. Firstly, the approach is sensitive to noise, such as oil stains, road marks, shadows, and bridge joints. Therefore, enhancing the approach's robustness in the next step would be helpful. Secondly, the current approach only focuses on binary classification in fixed patches. Hence, it is difficult to identify a specific defect in one step when different kinds of defects coexist in one image, especially for similar damage with different ROI (region of interest) sizes, such as potholes and cracks. The hierarchical ensemble learning and flexible region proposals are promising to solve this issue. Thirdly, the support examples should be representative across the overall items because different support sets will result in different performances in damage detection. However, it requires a combination of ML and domain knowledge. Hence, how to select the support examples needs further study.

Chapter 6 Point-cloud damage assessment and DT synchronization

Point clouds are widely used for structure inspection and can provide damage spatial information. However, how to update a DT with local damage based on point clouds has not been sufficiently studied. This research presents an efficient framework for assessing and DT synchronizing local damage on a planar surface using point clouds. The pipeline starts from damage detection via DeepLabV3+ on the pseudo grayscale images from the point depth. It avoids the drawbacks of image and point cloud fusion. The target point cloud is separated according to the detected damage. Then, it can be converted into a 3D binary matrix through voxelization and binarization, which is highly lightweight and can be losslessly compressed for DT synchronization. The framework is validated via two case studies, demonstrating that the proposed voxel-based method can be easily applied to real-world damage with non-convex geometry instead of convex-hull fitting; FE and BIM models can be updated automatically through the framework.

6.2 Proposed framework

The overall design of the proposed framework is shown in Figure 47, including data preparation, surface damage detection, point cloud processing, and DT synchronization. In data preparation, the target planar surface in the raw point cloud is calibrated through normal lines and rotation matrices. The surface damage detection is achieved through the state-of-the-art semantic segmentation model (i.e., DeepLabV3+) based on the pseudo grayscale image from the point depth. The point-cloud processing for damage spatial segmentation is achieved through voxelization and binarization. The result representing the spatial damage geometry is a highly lightweight 3D binary matrix and can be losslessly compressed for data transmission. Finally, the DT model synchronization (i.e., FE and BIM models) and damage volumetric assessment can be achieved automatically from the transmitted 3D binary matrix.

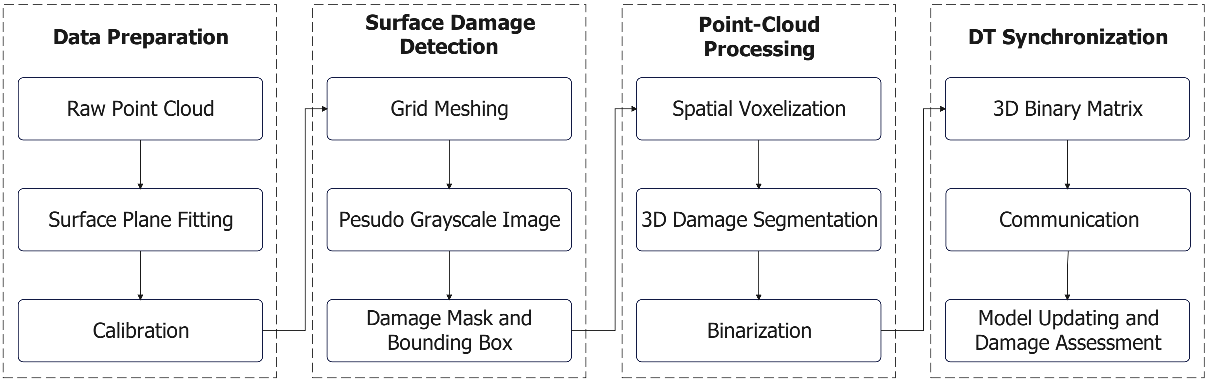


Figure 47 Proposed framework for 3D damage assessment and DT synchronization

6.3 Data preparation

In the beginning, the target surface plane in the point cloud can be fitted using the M-estimator Sample Consensus (MSAC) algorithm by finding a plane that has a maximum allowable distance from an inlier to it [243]. The MSAC algorithm is a variant of the Random Sample Consensus (RANSAC) algorithm, which can partially compensate for the undesirable effect of noise threshold selection [244]. Then, the point cloud can be calibrated by adjusting the surface normal line perpendicularly to the horizontal plane. The calibration can be achieved by multiplying the rotation matrix M, indicated in Eq. 30.

$$M = \begin{bmatrix} cos\,\beta & 0 & sin\,\beta \\ 0 & 1 & 0 \\ -\,sin\,\beta & 0 & cos\,\beta \end{bmatrix} \tag{30}$$

Where β is the pitch angle between the surface normal line and the z-axis.

This process can be illustrated by checking the verticality of a damaged RC column based on a point cloud, as shown in Figure 48. The column base upper surface is fitted using the MSAC algorithm and calibrated through the above rotation matrix. It is worth noting that the normal points on the target surface should be sufficiently more than the damage face points, which is required for surface plane fitting.

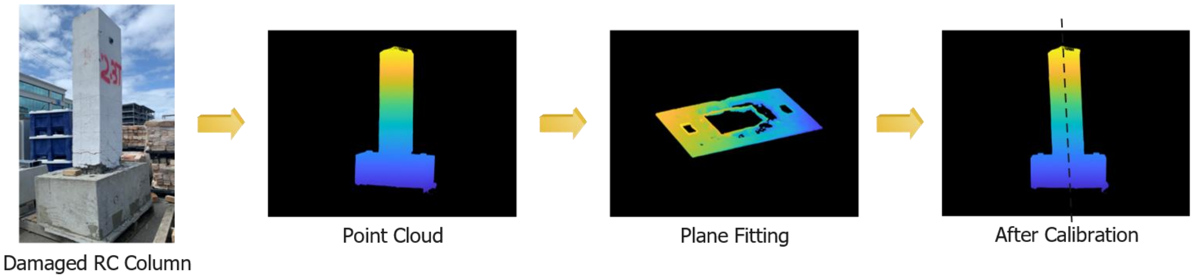


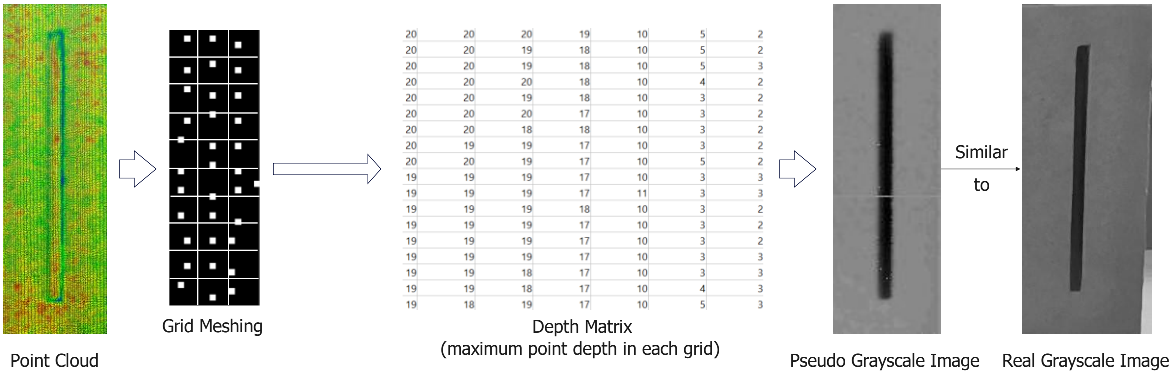
Figure 48 Plane fitting and calibration of point cloud to check damaged column verticality

6.4 Surface damage defection

6.4.1 Depth-to-grayscale transformation

In previous research [142], surface damage detection on a point cloud is achieved through image processing or DL on 2D images (usually RGB) and fusion with the point cloud. It requires extra photo acquisition with a built-in camera. If the photo cannot be taken under the same condition as the survey, i.e., from the same angle and distance, it will lead to complex coordinate transformation and potential errors. This work aims to solve this issue by achieving surface damage detection with pseudo-grayscale images entirely based on point-cloud depth information.

Assuming the deeper spot in the damaged space has lower luminance (i.e., lower grayscale level), the pseudo grayscale images can be generated through the pipeline shown in Figure 49. Here, the point cloud for the manual groove on a specimen is taken as an example. The resolution in grid meshing is considered consistent for the following voxelization and downstream tasks, determined by damage conditions and assessment requirements. For example, the evaluation for the building crack in the maintenance manuals [245]–[247] is at the millimetre level, so the survey for cracking is usually achieved with high-resolution TLS scanning, and the grid resolution is set as 1 mm. In contrast, the assessment for spalling, which is much broader than crack with more relaxed inspection standards, can be achieved using the iPhone LiDAR under 1 cm resolution for both volumetric quantification and FE model updating in the previous research [142].



20	20	20	19	10	5	2
20	20	19	18	10	5	2
20	20	19	18	10	5	3
20	20	20	18	10	4	2
20	20	19	18	10	3	2
20	20	20	17	10	3	2
20	20	18	18	10	3	2
20	19	19	17	10	3	2
20	20	19	17	10	5	2
19	19	19	17	10	3	3
19	19	19	17	11	3	3
19	19	19	18	10	3	2
19	19	19	17	10	3	2
19	19	19	17	10	3	2
19	19	19	17	10	3	3
19	19	18	17	10	3	3
19	19	18	17	10	4	3
19	18	19	17	10	5	3

Figure 49 Pseudo grayscale images derived from point-cloud depth information

Initially, grid meshing is applied on the point cloud to tighten the point sparsity. As seen in Figure 49, multiple points can be in a single grid, and the depth matrix is generated with the maximum point depth in the grid. Notably, this process will not lead to the loss of information related to the crack. In contrast, it will enhance the impact of the maximum depth in the grid because the deepest point represents the depth of a single grid.

Then, the depth-to-grayscale transformation for pseudo grayscale images is achieved through Eq. 31 – Eq. 33. d is the depth value for each element in the depth matrix; d_{th} is the depth threshold for normalization (i.e., $d = d_{th}, \forall d > d_{th}$); I_{mean} is the average grayscale level of the training image set I; D_{gray} is the generated grayscale level for each pixel in a pseudo-grayscale image.

$$D = 1 - \frac{d}{d_{th}} \tag{31}$$

$$K = \frac{I_{mean}}{D_{mean}} \tag{32}$$

$$D_{gray} = K \cdot D \tag{33}$$

Moreover, the ratio k between the minimum damage depth d'_{min} and the depth threshold d_{th} is an essential indicator for successful transformation, indicated in Eq.

34.

$$k = \frac{d'_{min}}{d_{th}} \tag{34}$$

Here, a realistic damage image set I [248] is utilized to train a DNN model for surface damage detection on the real grayscale images and then apply the model to the pseudo grayscale images. The pseudo-images are scaled to the same average grayscale level of the damage image set I (i.e., I_{mean}), which is beneficial for the model's performance. As the pseudo grayscale images are entirely generated from the point-cloud depth information, this method overcomes the angle and distance difference between the point cloud and the photo (see Figure 49), avoiding complex coordinate transformation and potential errors within data fusion.

6.4.2 DeepLabV3+ model

Surface damage detection on the generated pseudo grayscale images can be achieved via image processing such as OTSU’s method [249] through exhaustively searching the optimal threshold to maximize inter-class variance (Eq. 35) based on grayscale. The result can reflect the depth difference between the surface and the damaged areas. However, it cannot distinguish between natural damage and legitimate concave patterns, such as handcrafted holes, brick joints, and decorative textures. Hence, DL is utilized to solve this issue for surface damage detection.

$$\sigma_w^2(t) = \omega_0(t)\sigma_0^2(t) + \omega_1(t)\sigma_1^2(t) \tag{35}$$

Where ω_0 and ω_1 are the probabilities of the two classes (i.e., background and damage areas) separated by a threshold t; σ_0^2 and σ_1^2 are variances of these two classes.

This work uses the state-of-the-art semantic segmentation model DeepLabV3+, which combines the Atrous Spatial Pyramid Pooling benefits and the Encoder-Decoder architecture for surface damage detection and segmentation on grayscale images.

Here, the supervised learning approach for crack detection is an example and can be easily extended to detect other surface damages (such as spalling) by using the annotated corresponding images to train the model.

The DeepLabV3+ model is trained on a public crack image set [248], which includes 9584 crack and 1411 non-crack images. The images are all transformed into grayscale pictures and split into the training, validation, and test sets (i.e., 80%:10%10%). The pre-trained MobileNet-v2 based on the PASCAL VOC dataset [250] is employed as the backbone for feature extraction. The training condition is shown in Table 10. The training loss and MIoU are presented in Figure 50. The performance on the test set is evaluated using mean Intersection over Union (MIoU) and mean Pixel Accuracy (MPA), which are 80% and 88%, respectively. It demonstrates that the trained DeepLabV3+ has excellent performance for crack detection and segmentation on real grayscale images. Moreover, the model can distinguish between natural cracks and legitimate concave patterns such as handcrafted holes and brick joints.

Table 10 Model training condition

Architecture	Input	Split	Epochs	Batch Size	Learning Rate
DeepLabV3+	448×448	80%:10%:10%	100	8	0.00005

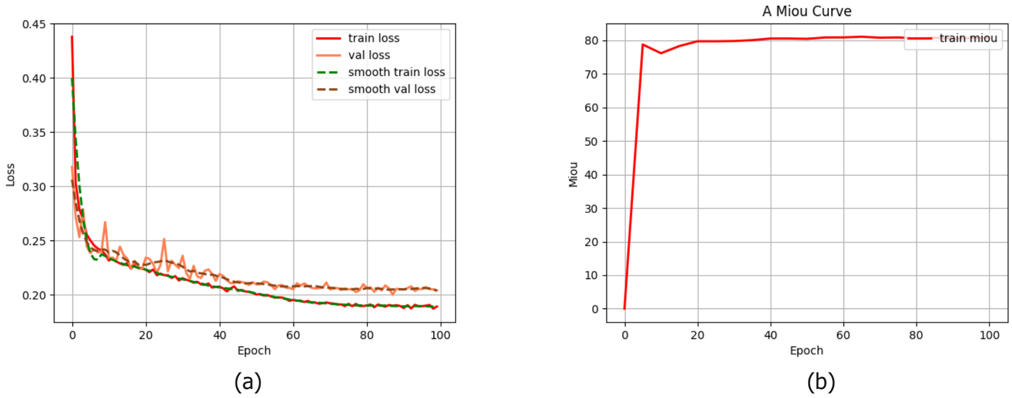


Figure 50 (a) training loss and validation loss; (b) training MIoU

Four specimens created in the lab with variant crack widths and depths are utilized to test the model performance on the pseudo grayscale images, as shown in Figure 51.

The RealSense LiDAR Camera L515 is used for crack scanning to generate point clouds. The real RGB images are derived from the RGB information for each point using the built-in camera, and its grayscale images are calculated through Eq. 36. The pseudo grayscale images are generated through the pipeline in Figure 49 with a resolution of 1 mm and $d_{th} = d_{max}$.

$$\text{Gray} = 0.2989 * R + 0.5870 * G + 0.1140 * B \tag{36}$$

The proposed approach is tested by comparing the model performance on the pseudo grayscale images with the real grayscale images, and the latter is taken as the ground truth for segmentation. The model performance is evaluated using the MIoU (Eq. 37) and the MPA (Eq. 38) for binary segmentation, i.e., crack (positive) and background (negative) pixels. IoU_{pos} and IoU_{neg} denote positive and negative Intersection of Union; P_{pos} and P_{neg} represent positive and negative precision; TP, FP, TN, and FN denote true positive, false positive, true negative, and false negative pixels, respectively.

$$MIoU = \frac{IoU_{pos} + IoU_{neg}}{2} = \frac{\frac{TP}{TP + FP + FN} + \frac{FN}{TN + FN + FP}}{2} \tag{37}$$

$$MPA = \frac{P_{pos} + P_{neg}}{2} = \frac{\frac{TP}{TP + FP} + \frac{TN}{TN + FN}}{2} \tag{38}$$

The segmentation results through the trained DeepLabV3+ model on both real and pseudo-grayscale images are demonstrated in Figure 51. Here, the crack depth is the vertical depth detected by LiDAR. As can be seen, in the experiment for the cracking with width > 5 mm and depth > 6 mm, the trained DeepLabV3+ model can achieve excellent crack detection and segmentation performance (i.e., $MIoU = 84.60\%$, $MPA = 97.20\%$) on the pseudo grayscale images, which are entirely derived from the point depth. The failure on the fourth beam is because the crack is too narrow, and the LiDAR cannot obtain sufficient points within the cracking space due to occlusion.

Notably, although this approach is illustrated with crack detection and segmentation, it is also available to detect other volumetric damages on a planar surface, such as spalling.

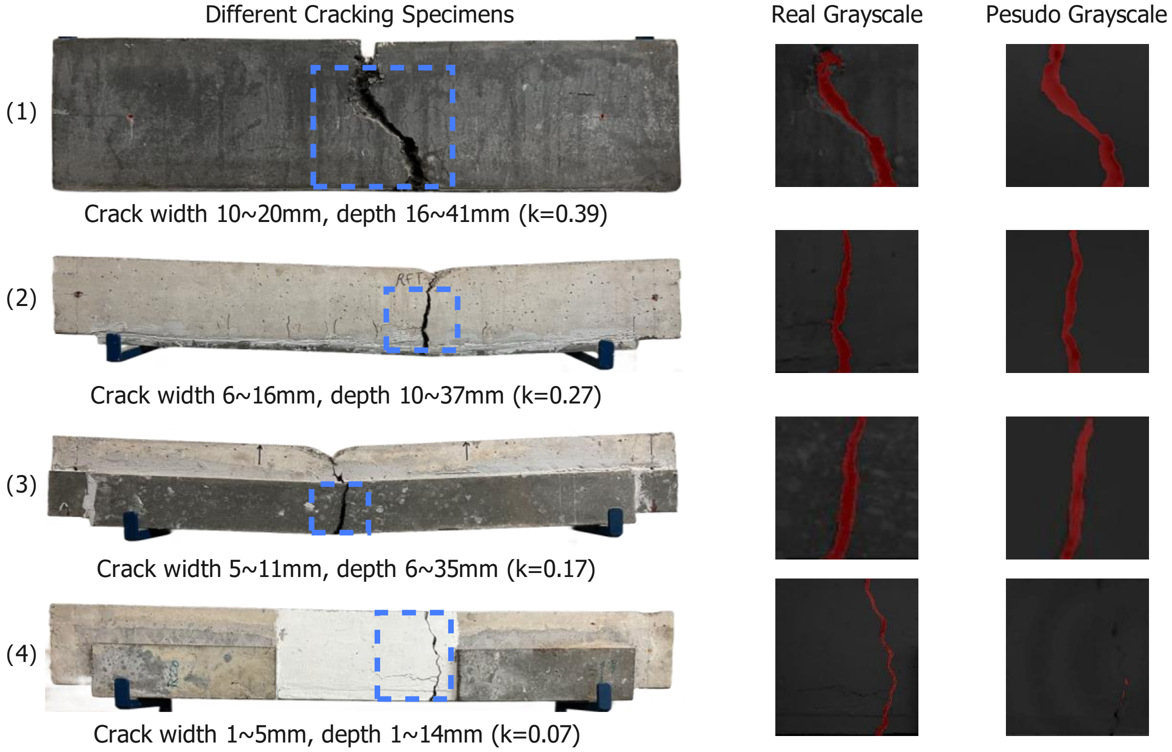

Different Cracking Specimens
Real Grayscale
Pesudo Grayscale
(1)
Crack width 10~20mm, depth 16~41mm (k=0.39)
(2)
Crack width 6~16mm, depth 10~37mm (k=0.27)
(3)
Crack width 5~11mm, depth 6~35mm (k=0.17)
(4)
Crack width 1~5mm, depth 1~14mm (k=0.07)

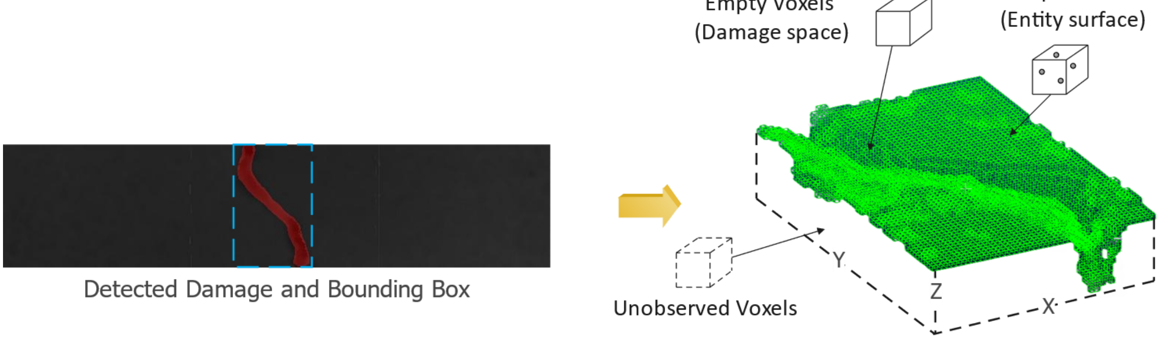
Figure 51 Damage segmentation using DeepLabV3+ model on the pseudo grayscale images

The experiment demonstrates that when structural damages satisfy certain conditions (e.g., cracking width > 5mm and $k > 0.17$ in the experiment), a pre-trained DNN model from the real grayscale damage images with annotation can achieve excellent performance for surface damage detection and segmentation on the pseudo grayscale images, which are entirely derived from the point depth with appropriate resolution. This approach avoids the drawbacks of image and point cloud fusion, such as extra photo acquisition, complex coordinate transformation, and potential data fusion errors. Notably, the resolution selection is affected by the survey equipment, such as the 1 cm resolution with iPhone LiDAR in the previous research [142]. The model performance may degrade when using a relatively low resolution to generate pseudo-grayscale images. At that moment, a new model would be required based on the reduced training images with the corresponding resolution. After training, the model can also distinguish natural structural damages from legitimate concave patterns.

6.5 Spatial damage segmentation

6.5.1 Spatial voxelization

After surface damage detection and segmentation, the damage can be masked with a bounding box. Then, the point cloud section involving target damage can be separated as a cuboid according to the bounding box from the surface until the maximum depth along the depth direction, i.e., the z-axis, as shown in Figure 52. The separated cuboid section can be fully voxelized with an appropriate resolution, which should be consistent with the grid resolution in the previous stage and suitable in downstream tasks for damage assessment. As seen in Figure 52, the cuboid after voxelization includes three different types of voxels, i.e., (1) the empty voxels in the damage space, (2) the occupied voxels by the entity surface and damage face, (3) the unobserved voxels due to occlusion. The benefit of doing this is that the enclosed damage space and the entity can be distinguished using the status of each voxel, i.e., the empty and the occupied voxels. Here, the voxel status is defined as occupied even if only a single point is included, so the voxelization will not lead to losing the point information. By contrast, it will enlarge the single-point impact, which is beneficial for assessment reliability.

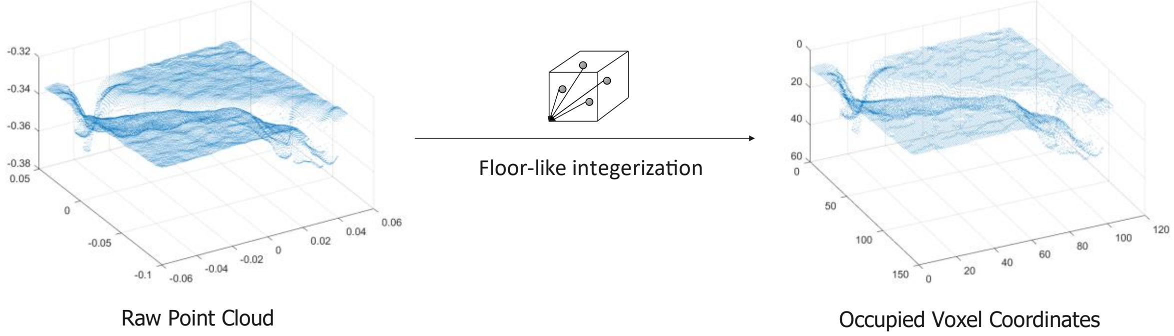


Detected Damage and Bounding Box

Damage Section and Spatial Voxelization

Figure 52 Separated cuboid damage section and spatial voxelization

In principle, voxelization can be implemented through the Octree [251]. The observed occupied voxels on the surface can be easily obtained using floor-like integerization for point coordinates according to the resolution, as shown in Figure 53. Here, the integer coordinate (x, y, z) of each node at the corner corresponds to the position of each occupied voxel in the cuboid.



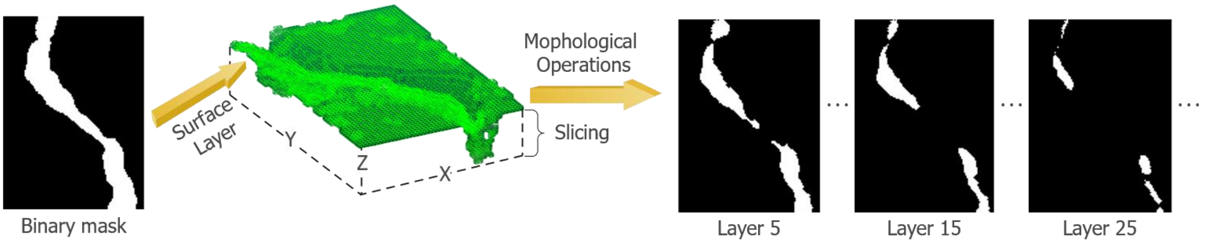
Raw Point Cloud

Occupied Voxel Coordinates

Figure 53 Voxelization through floor-like integerization

6.5.2 Binarization and morphological operation

The binary mask for the detected damage on the surface is shown in Figure 54, including the damage area (white pixels – 1) and the background area (black pixels – 0). It can be utilized as the damage mask directly on the surface layer of the separated cuboid damage section when grid meshing and voxelization have the same resolution. Otherwise, the mask needs to be resized.



Binary mask

Layer 5

Layer 15

Layer 25

Figure 54 Surface binary mask and following morphological operations

After assigning occupied voxels with 0 and empty voxels (including unobserved voxels) with 1, each layer (or slice) of the voxelized cuboid along the depth direction can be taken as a binary image. Then, the empty voxels, occupied voxels, and unobserved voxels in the following layers along the depth direction are updated through morphological operations, as shown in the algorithm below. The Hadamard product enables the damage area to shrink when newly occupied voxels arise in the current layer, and image closing can remove the outlier pixels enclosed in the damage area. Finally, the result is a highly lightweight 3D binary matrix representing the damage spatial geometry.

```
Algorithm 1 Binarization and morphological operation
Input: Voxelized cuboid damage section and structural element $s$
Output: 3D binary matrix to represent damage spatial geometry
Binary mask $M_0 \leftarrow$ $surface\ damage\ segmentation$ ($damage$ 1, $background$ 0)
while $i \le d_{max}$ ($maximal\ depth$) do
    $M_i = M_{i-1} \odot M_i (Hadamard\ product)$
    $M_i = M_i \oplus s$ ($Dilation$)
    $M_i = M_i \ominus s$ ($Erosion$)
end
return $M \leftarrow$ concatenate ($M_1, M_2, \ldots M_{d_{max}}$)
```

Notably, this processing method relies on the detected damage face points and will not lead to the loss of the damage space. For example, if the equipment cannot perceive the damage face in a few layers, these layers will inherit the damage mask of the previous layer. Hence, the damage area in each layer will only change as the new occupied voxels arise, i.e., new damage face points are detected. It is beneficial for assessment reliability based on the perceived point cloud.

6.6 DT synchronization and damage assessment

6.6.1 Data compression and transmission

The transmission data includes the generated 3D binary matrix representing the damage spatial geometry and the corner point coordinates of the segmented cuboid on the surface for geo-referencing. The 3D binary matrix is much lighter than the original point cloud and can be further compressed through the lossless RLE before transmission, as shown in Figure 55. The compression efficiency can be evaluated with compression ratio (CR), indicated in Eq. 36.

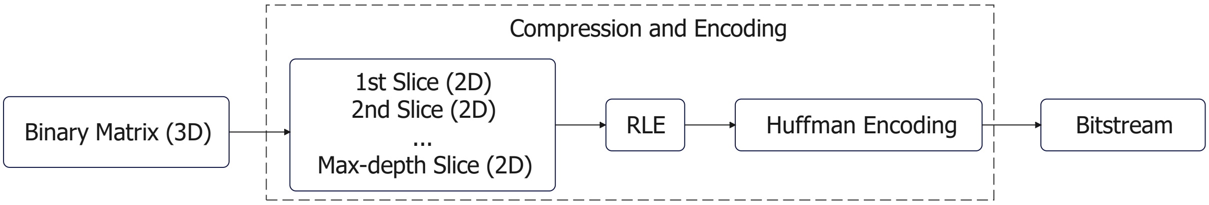


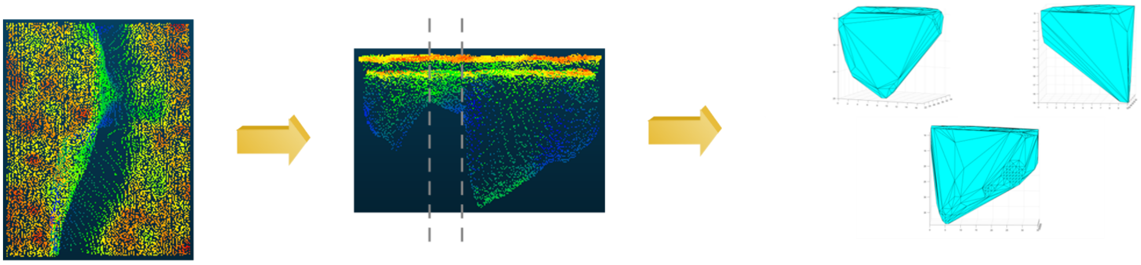
Figure 55 3D Binary matrix compression and transmission

$$Compression\ Ratio = \frac{Uncompressed\ Size}{Compressed\ Size} \tag{39}$$

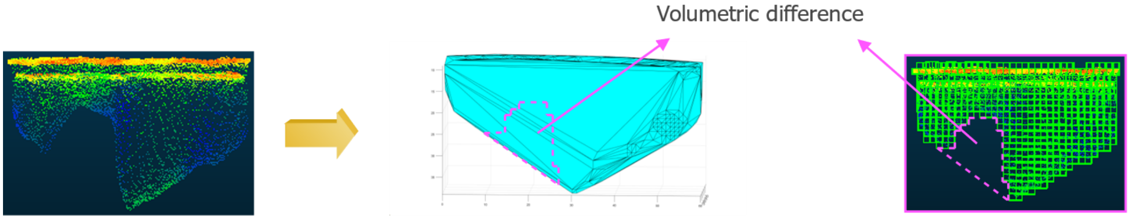
6.6.2 Damage volumetric assessment

The damage volume assessment aims to evaluate the residual bearing capacity of the damaged structure by quantifying the damaged void space. The latest research [142] uses convex-hull fitting to achieve damage volumetric quantification, but this method requires handcrafted cutting for each convex component. Otherwise, it will exaggerate the damage volume. However, elaborate manual cutting is time-consuming and is not always available for real-world structural damage with complicated non-convex geometry. Hence, voxel-based fitting is proposed to solve this issue in this work.

A point cloud for cracking is shown in Figure 56. Here, it assumes the crack is wide enough, and the survey provides sufficient points for damage spatial assessment. As can be seen, the damage space is a non-convex geometry, which can be manually separated into three convex polyhedrons for convex-hull fitting, as shown in Figure 56 (a). According to previous research [142], the sum of these three volumes can be taken as the ground truth. Figure 56 (b) shows the convex-fitting result for the intact point cloud without manual separation, and Figure 56 (c) demonstrates the voxel-based fitting result. As can be seen, the convex-hull fitting without manual separation tends to exaggerate the damage volume with the purple-marked volume, and the voxel-based fitting is much closer to the ground truth. In practice, the voxel-based fitting volume can be easily obtained by summing up the value of the transmitted 3D binary matrix (i.e., empty voxel – 1 and occupied voxel – 0) to figure out the voxel amount and then multiplying it with the volume of the unit voxel.



(a) Ground truth



(b) Convex-hull fitting without manual cutting

(c) Voxel-based fitting

Figure 56 Comparison of convex-hull fitting and voxel-based fitting: (a) ground truth; (b) convex-hull fitting without manual separation; (c) voxel-based fitting

6.6.3 Model synchronization

6.6.3.1 FE model geometric updating

Here, the element for FEA has the equivalent resolution of grid meshing and voxelization in Section 3.4 and Section 3.5. The latest research [142] demonstrates that the FE model can be updated geometrically by removing the elements less than the damage depth, but the algorithm is based on loop iteration. In principle, with the generated 3D binary matrix, the more efficient method to update the FE geometric model is to apply the element-wise product on the corresponding cuboid (after geo-referencing) in the FE model, as shown in Figure 57. Its time complexity is $O(n)$, where $n = l \times w \times d$ (l, w, and d are numbers of elements in the matrix along length, width, and depth, respectively), and space complexity is $O(1)$.

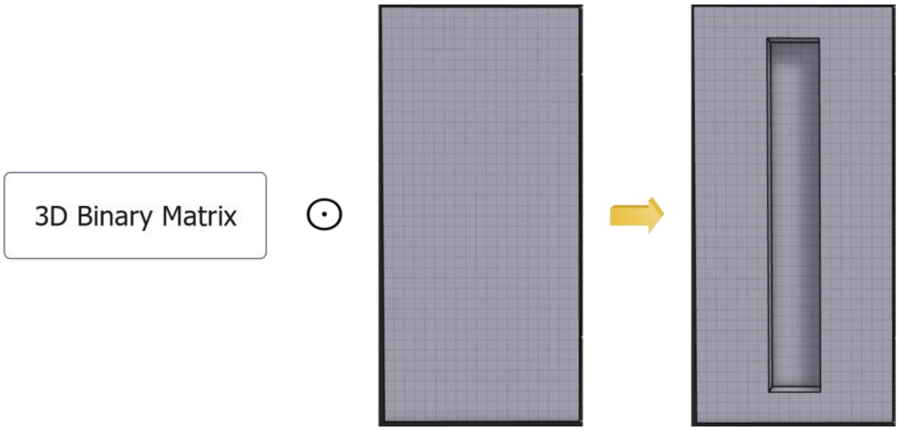


Figure 57 Designed FE model geometric updating through elementwise product

However, most commercial FE software cannot support this manipulation, and the elements are usually denoted with indices rather than a matrix, such as Abaqus and Ansys. Hence, the empty voxel coordinates are retrieved by querying with the matrix element equal to 1 and then used to generate the damage elements' indices. Finally, the FE model can be updated geometrically by deleting the corresponding elements or setting them invalid for the calculation. The pseudocode of the process is shown below.

```
Algorithm 2 FE model geometric updating
Input: 3D binary matrix representing damage spatial geometry
Output: Updated FE geometry model
Empty voxel coordinate set $\{X, Y, Z\}$ by querying $M = 1$ (empty =1, occupied = 0)
while $\{x_i, y_i, z_i\}$ in $\{X, Y, Z\}$ do
    Void element $id \leftarrow \{\mathrm{x}_i, y_i, z_i\}$ and $\{\mathrm{x}_0, y_0, z_0\}$
    Append $id$ into void element list $L$
end
Delete the elements or set them invalid according to $L$
```

6.6.3.2 BIM model geometric and semantic updating

The damage contour in each layer (or slice) of the transmitted 3D binary matrix can be fitted with an appropriate shape. For example, the circles or ellipses fitting for spalling can indicate each layer's centroid and radius (including major and minor radius), which is helpful for damage recording and assessment. The 3D damage geometry can be constructed automatically from the transmitted matrix through commercial software like Dynamo. Then, the BIM model can be further updated with the damage geometry following the pipeline shown in Figure 58 according to previous research [150], [252], [253]. The fitted damage polyhedron can be taken as an entity and assigned with *IfcVoidingFeature*, a modification of an element to reduce its volume. After setting its parameter *PredefinedType* to *CUTOUT*, the damage geometry can be subtracted from the intact component via the *DamagedGeometryCutout* relationship.

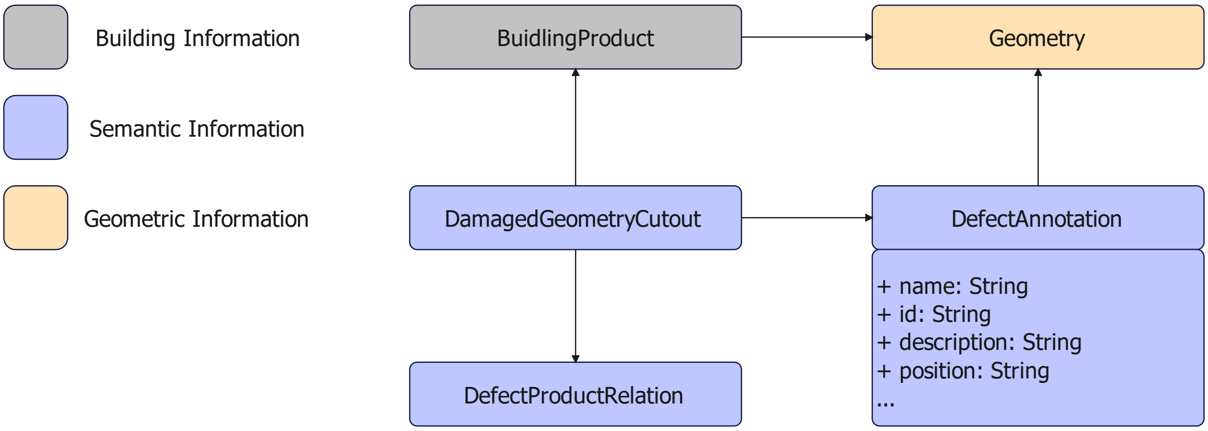


Figure 58 BIM model updating with the local damage geometry [253]

6.7 Experiment preparation

A public TLS point cloud dataset [141], [254] for synthetic grooves and real-world cracking is utilized for framework validation. The synthetic grooves are created on a specimen with different widths (from 1mm to 10mm), and the cracking happens on a building wall, as shown in Figure 59. The survey uses a high-quality laser scanner (Z+F IMAGER® 5016) at 5m with an incidence angle of 0°. The scanner's distance resolution is 0.1 mm, and the linearity error is less than 1 mm + 10 ppm/m. The blue rectangles indicate the selected point clouds for the experiment. CloudCompare and MATLAB are utilized for point-cloud processing. Dynamo is used to reconstruct the 3D damage geometry automatically, and Abaqus is used for updating the FE geometric model.

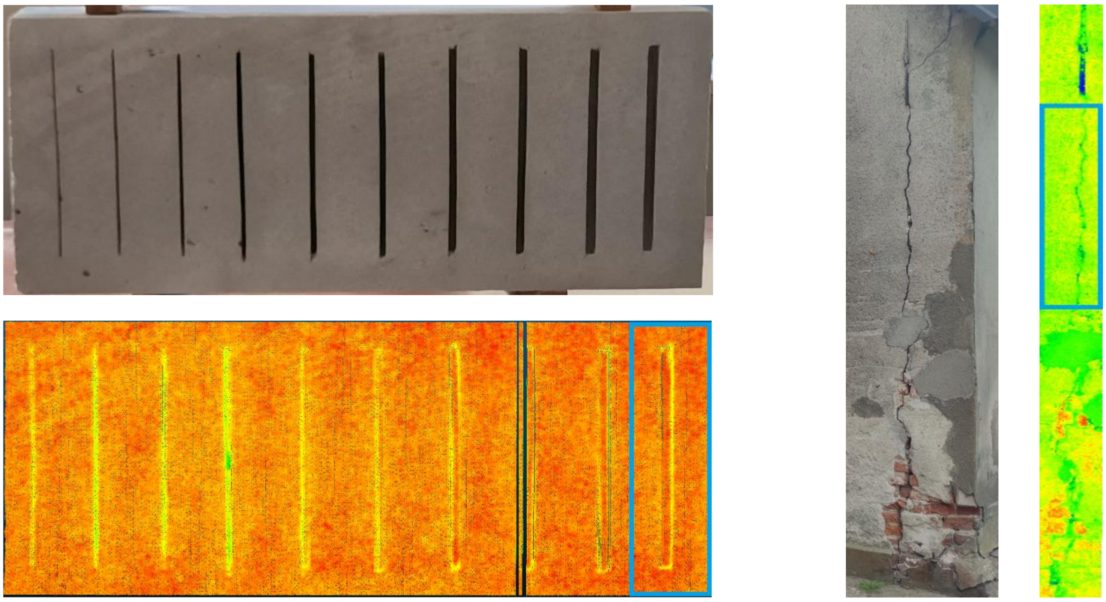

(a) (b)

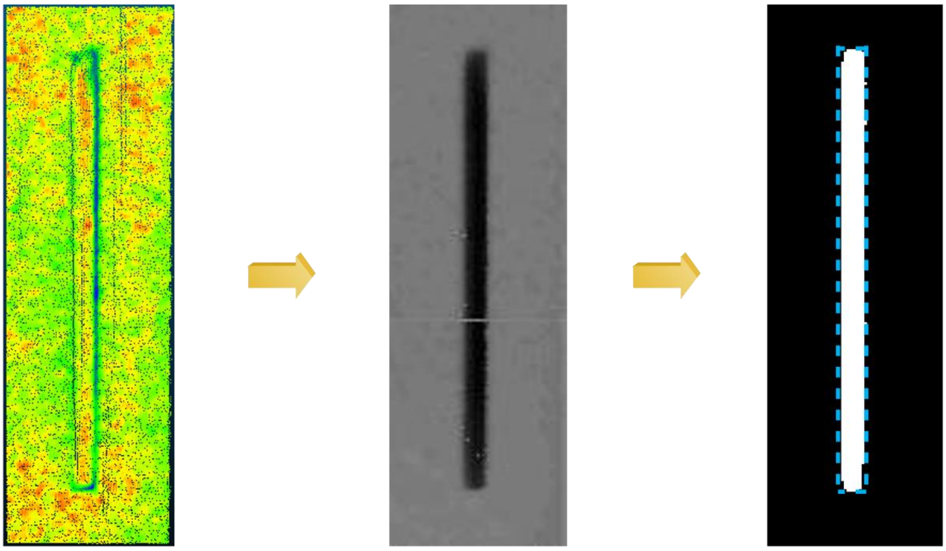
Figure 59 (a) Synthetic grooves; (b) Real-world building crack

6.8 Case study 1 – synthetic groove

6.8.1 Groove detection and spatial segmentation

As the wider groove has a better scanning performance for its internal space, the biggest groove with a 10 mm width is adopted for the experiment, shown as blue-marked in Figure 59 (1). Previous research indicates that its spatial geometry can be described effectively using the point cloud for damage detection[141]. The target groove is initially cropped from the point cloud, and the surface plane is fitted using the MSAC. The pitch angle β of the fitted surface is -0.0218 rad, so the point cloud can be calibrated through Eq. 30 around the y-axis by β, to make sure the fitted surface plane is horizontal.

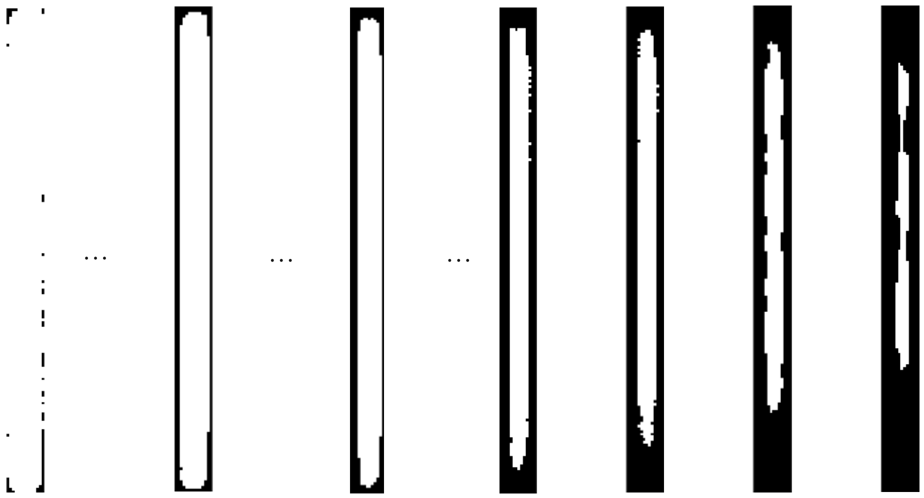
The pseudo grayscale image for the cropped section is generated based on the point depth information through the pipeline shown in Figure 49, with an average grayscale of 124. The grid resolution is 1 mm. Because the trained DeepLabV3+ model cannot recognize synthetic damage (like a groove), the OTSU thresholding method is adopted here for groove segmentation on the surface. The groove region, i.e., the mask and the bounding box, can be generated, as shown in Figure 60.



Point Cloud of Synthetic Groove Pseudo Grayscale Image Mask and Bounding Box

Figure 60 Groove detection on a pseudo grayscale image

Then, a 3D cuboid involving the groove is separated according to the bounding box from the surface plane until the maximum depth. The cuboid can be voxelized spatially with the same resolution as grid meshing in the above stage, i.e., 1 mm. The occupied voxels by the entity surface and the damage face are set to 0; the empty voxels (including unobserved voxels) are set to 1. Subsequently, the spatial segmentation for the groove is achieved through Algorithm 1 in Section 3.5.1. The voxelized cuboid's layers (or slices) can be shown in Figure 61, where empty voxels are white and occupied voxels are black. Finally, a 3D binary matrix M $(200 \times 35 \times 21)$ representing the groove spatial geometry is generated as a ".mat" file.



Surface layer 10th layer 15th 18th 19th 20th 21st

Figure 61 Each layer (or slice) for the groove in the voxelized section

6.8.2 DT synchronization

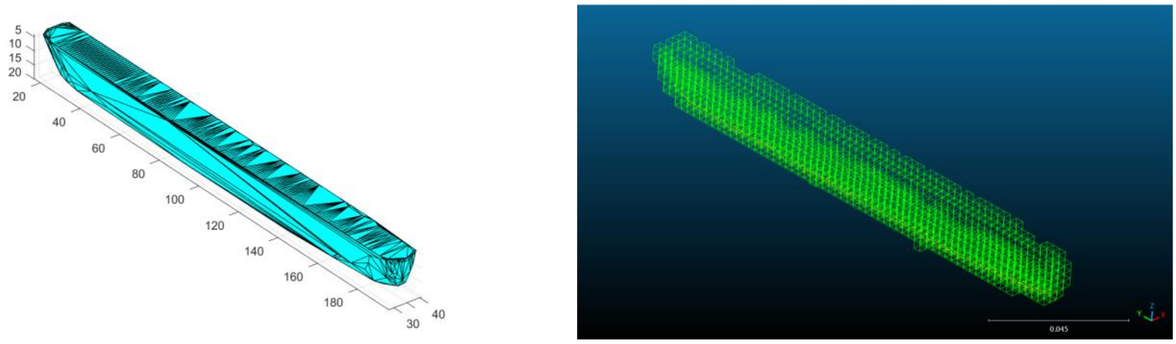
6.8.2.1 Data compression and transmission

The blue-marked point cloud in Figure 59 (1) has 9.5428×10^4 points and is saved as a ".pcd" file of 2.18 MB. In contrast, the generated 3D binary matrix is saved as a ".mat" file of only 3KB, and it can be further compressed through RLE to 1148 bytes. The compression rate (CR) reaches 61%. Meanwhile, the x-y coordinates (m) of the bounding box diagonal corner points, i.e., (0.029, 0.197) and (0.042, 0.018), are utilized for geo-referencing of the target groove section on the surface plane. It demonstrates that the proposed framework can enable highly efficient data transmission for DT synchronization with the target groove. The generated data through voxelization and binarization is significantly lighter than the original point cloud, which can support the timely as-is model updating during the survey.

6.8.2.2 Groove volumetric assessment

As the target vertical groove is a simply convex geometry, the convex-hull fitting method in the previous research [142] can be implemented directly on the groove point cloud without handcrafted separation as a benchmark, and its estimated volume is

25687 mm^3. The voxel-based fitting assessment can be achieved by summing up the Boolean values of the transmitted 3D binary matrix (i.e., the unit voxel is 1 mm^3), and the volume estimation is 25231 mm^3. As can be seen, the result difference between the two fitting methods is less than 2% for the target groove, demonstrating that the proposed voxel-based fitting is acceptable for the volumetric assessment of the 3D damage with a convex geometry based on appropriate resolution. Both fitting methods' performance is presented in Figure 62.



(a) Convex-hull Fitting (b) Voxel-based Fitting

Figure 62 Groove volumetric assessment with convex-hull and voxel-based fitting

6.8.2.3 FE model geometric updating

A model for FEA is established in Abaqus with the equivalent resolution of voxelization. Initially, the target groove section (corresponding to the transmitted 3D binary matrix) can be localized in the FE model using the diagonal corner point coordinates. Then, the ID list of the empty elements can be obtained by retrieving the matrix element that equals 1. Finally, the FE model can be updated automatically by deleting the open elements in the groove using the Python-based script, as shown in Figure 63. It demonstrates that the proposed framework can automatically enable FE model geometric updating with the 3D local damage.

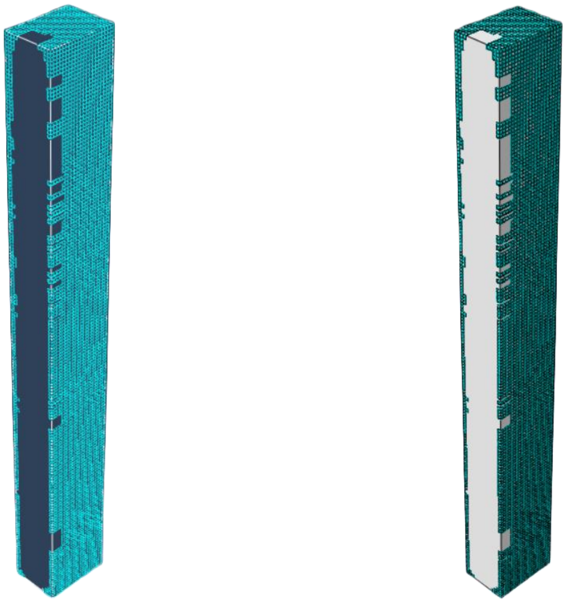
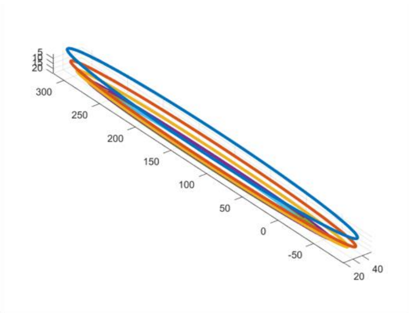

(a) Groove in hidden mode (b) Groove in shaded mode

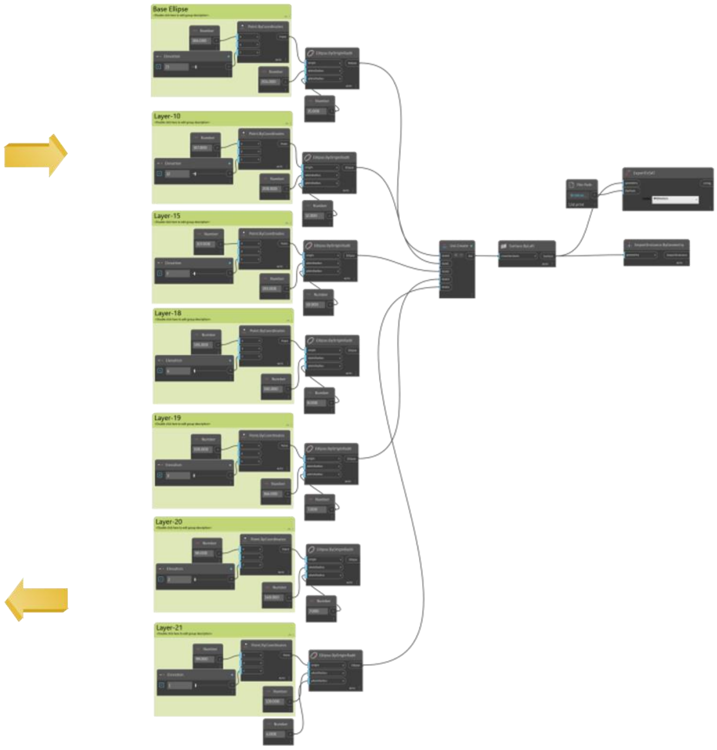
Figure 63 FE model geometric updating for synthetic groove

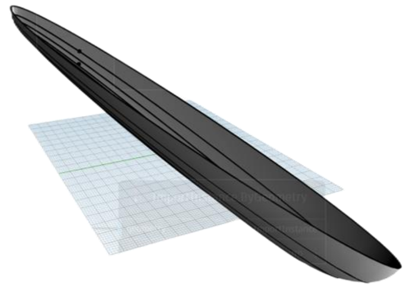
6.8.2.4 BIM model updating

The groove contour in each layer can be fitted with ellipses based on centroids, major and minor axes, as shown in Figure 64 (a), indicating the critical damage features, such as location, width, and length. The fitted ellipses can be generated in Dynamo using *Ellipse.ByOriginRadii* through visual programming, as shown in Figure 64 (b). Finally, the groove geometry can be reconstructed, as shown in Figure 64 (c).



(a) Fitting with ellipses





(c) Reconstructed groove geometry

(b) Visual programming

Figure 64 BIM model geometric updating for synthetic groove

Furthermore, the groove component can be taken as an entity with *IfcVoidingFeature,* and the parameter *PredefinedType is* set to *CUTOUT*. Then, the groove geometry can be subtracted from the intact *BuildingProduct* based on the *DamagedGeometryCutout* relationship. Consequently, *BuildingProduct* will point to the specimen with *componentGeometry*, and *DamagedGeometryCutout* will refer to the groove with *damagedGeometry*. Finally, *DefectAnnotation* (including name, id, description, position, etc.) can be associated with the groove component to update the semantic information in the BIM model. The complete pipeline is shown in Figure 58.

6.9 Case study 2 – real-world building crack

6.9.1 Crack detection and spatial segmentation

In the second case, the framework is tested on the point cloud of a real-world building crack, as shown in Figure 59 (2). The experiment adopts the blue-marked region, with the cracking width from 11 mm to 30 mm. As the high-resolution TLS scanning can provide sufficient spatial information for the synthetic groove with a width of 10 mm (see case 1), the point-cloud data under the same survey condition (i.e., the same equipment, angle, and distance) for the crack greater than 11 mm is available for damage spatial assessment. The surface plane is fitted with the MSAC, and the point-cloud calibration is achieved through Eq. 30 with $\beta = 0.0023\,rad$. The pseudo grayscale image is entirely derived from the point depth with a grid resolution of 1 mm and utilized for the crack segmentation on the surface through the pre-trained DeepLabV3+ model. The crack mask and bounding box are shown in Figure 65.

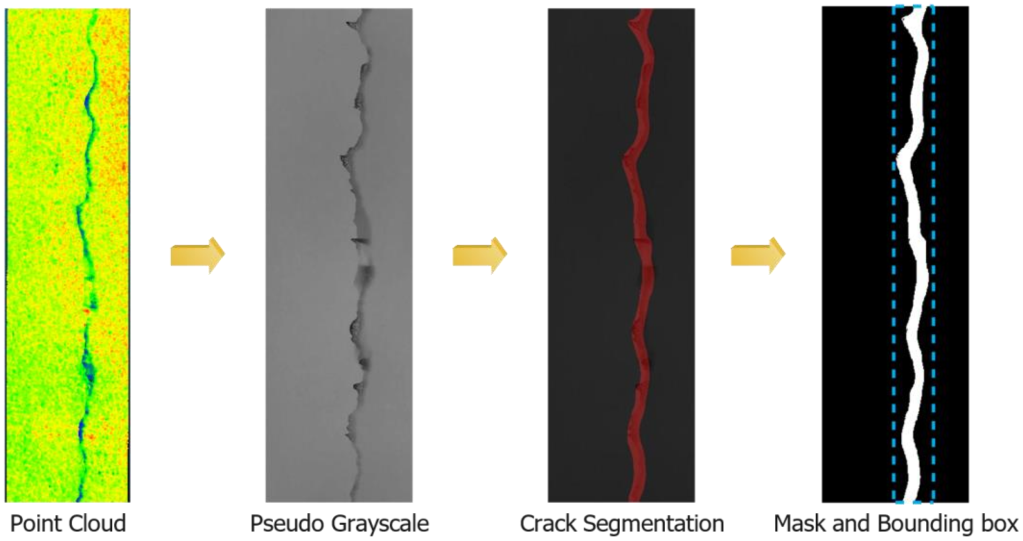


Figure 65 Real-world crack detection with pseudo grayscale image via DeepLabV3+

The 3D crack section (i.e., cuboid) is separated according to the bounding box from the surface plane until the maximum depth. The cuboid is voxelized spatially at a resolution of 1 mm. The occupied voxels by the entity surface and the damage face are set to 0; the empty voxels (including unobserved voxels) are set to 1. The spatial segmentation for the crack can be achieved through the algorithm in Section 3.5.1. The voxelized cuboid's layers (or slices) can be shown in Figure 66, where empty voxels are white and occupied voxels are black. Finally, a 3D binary matrix $M\ (850 \times 55 \times 56)$ representing the crack spatial geometry can be generated as a ".mat" file.

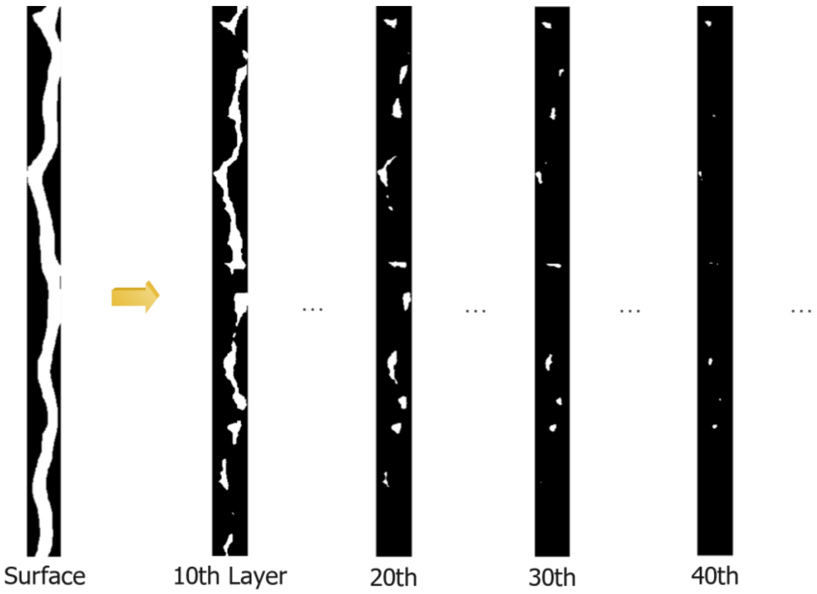


Figure 66 Each layer (or slice) for real-world cracking in the voxelized section

6.9.2 DT synchronization

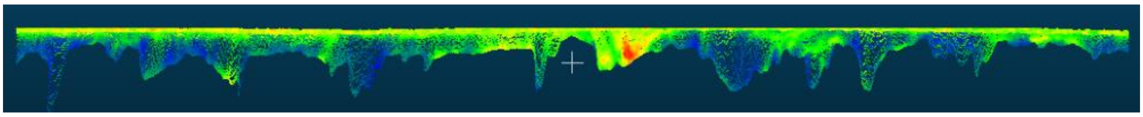
6.9.2.1 Data compression and transmission

The original point cloud for the target crack section in Figure 59 (2) has 2.141447×10^6 points and is saved as a ".pcd" file of 57.1 MB. In contrast, the generated 3D binary matrix is a ".mat" file of only 63KB, and it can be further compressed through lossless RLE until 19.3 KB. The compression rate (CR) reaches 69.37%. Meanwhile, the x-y coordinates (m) of the bounding-box diagonal corner points, i.e., (0.129, 0) and (0.184, 0.183), are utilized for geo-referencing of the target crack section on the surface. It demonstrates that the proposed framework can enable highly efficient data transmission for DT synchronization with the real-world building crack. The transmitted data is significantly lighter than the original point cloud, which can support the as-is model updating during the survey.

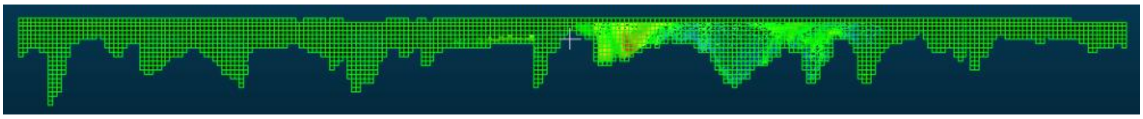
6.9.2.2 Damage volumetric assessment

The target real-world building crack is a complicated non-convex geometry, shown in Figure 67 (1), so it is impractical to apply elaborate handcrafted separation for each convex component to achieve volumetric assessment through the convex-hull fitting.

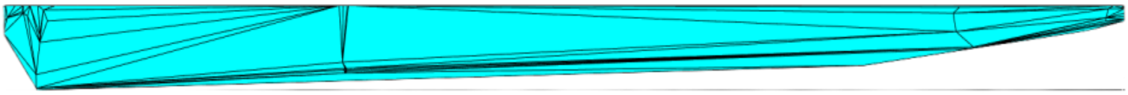
In contrast, the voxel-based fitting can be easily applied for the damage volumetric assessment in this situation. As case study 1 has demonstrated the effectiveness of the voxel-based method for a convex geometry, the voxel-based fitting with an appropriate resolution can be closer to the ground truth than the convex-hull fitting for the intact point cloud without manual cutting, as shown in Figure 67 (2) and (3). In practice, the voxel-based volumetric assessment can be achieved by summing up the Boolean values of the transmitted 3D binary matrix (i.e., the unit voxel is 1mm^3), and its result is 4.0652×10^5 mm^3. This method can also be extended for volumetric assessment of other 3D damages having non-convex geometry, such as the spalling in Figure 7. It demonstrates that the proposed voxel-based method can be easily applied for volumetric assessment of real-world damage having a non-convex geometry instead of convex-hull fitting.



(a) Target real-world building crack



(b) Voxel-based fitting for volumetric assessment

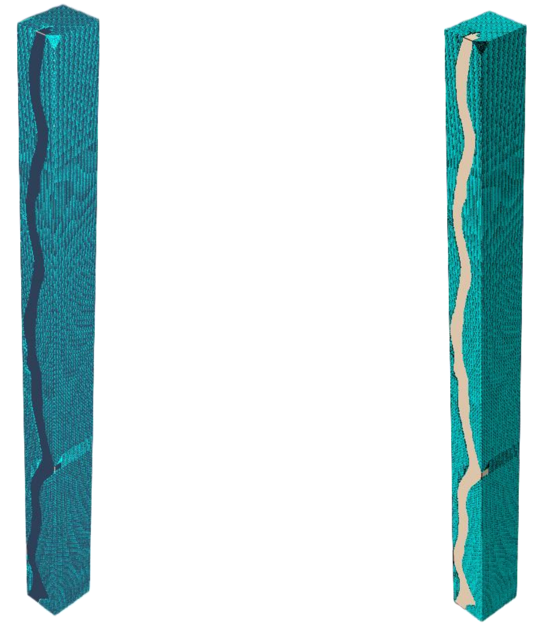


(c) Convex-hull fitting without manual separation

Figure 67 Volumetric assessment for the real-world building crack

6.9.2.3 Model updating

Like case 1, an FE model can be established in Abaqus with the equivalent resolution of voxelization. The separated crack section (i.e., a cuboid) can be localized using the diagonal corner point coordinates in the FE model. Then, the ID list of the empty element can be obtained by retrieving the matrix element that equals 1. Finally, as shown in Figure 68, the FE model can be updated automatically by deleting the open elements in the cracking space through a Python-based script. It demonstrates that the proposed framework can efficiently and automatically update the FE geometric model for real-world building cracks based on the point cloud.



(a) Crack in hidden mode (b) Crack in shaded mode

Figure 68 FE model geometrically updating for real-world cracking

Furthermore, the crack contour in each layer can be fitted using a bounding box, as shown in Figure 69, to indicate the cracking location and skeleton. Then, the crack spatial geometry can be generated by meshing the bounding boxes across different layers. Finally, the BIM model can be updated geometrically and semantically according to Section 6.6.3.2. The proposed framework for FE and BIM model updating with local damage is also available for other 3D structural damage, such as concrete spalling.

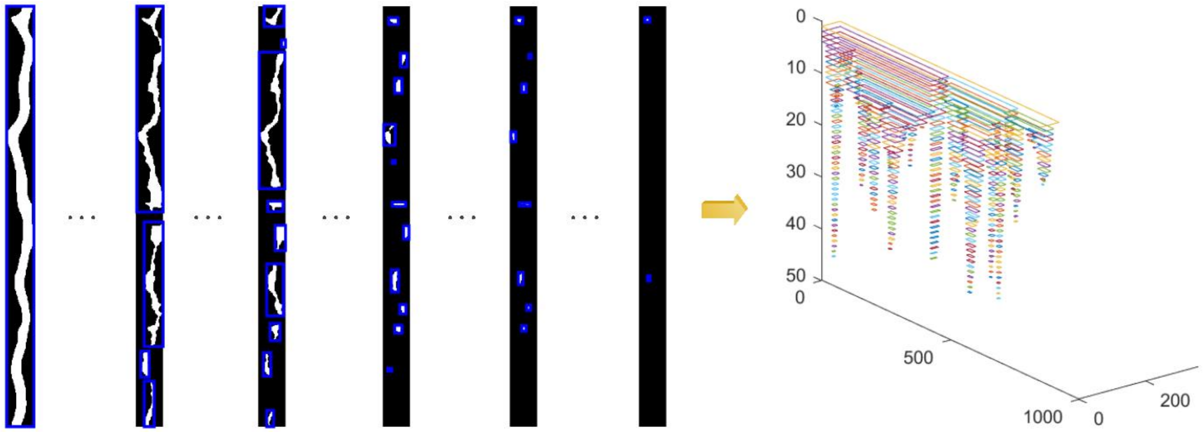


Figure 70 Crack contour in each layer fitted with bounding box

6.10 Discussion

Although the proposed framework performs excellently on synthetic and real-world spatial damage in the case studies, it still has some limitations. For example, the accessibility of the target from the LiDAR plays an important role in the proposed methodology while capturing the point cloud with the required interpoint spacing, which is crucial to the performance of damage assessment and reality modelling. This study keeps the scanner 5 m apart from the target and perpendicular to the planner surface without any angles in the experiment. However, this configuration may not always be practical for a survey of damaged structures if they cannot be accessible like this. Hence, it is necessary to discuss the ambient and intrinsic factors affecting the proposed methodology's performance, summarising as distance, angle, and edge effect (or laser beam size).

The distance from the scanner to the target influences measurement accuracy and point cloud density. As the distance grows, the measurement accuracy declines, so it is necessary to adapt the grid and voxel resolution accordingly. Although the accuracy remains acceptable within a specific distance range determined by the equipment, increased distance leads to a decrease in point density. This results in more empty grids or voxels without enclosed points, consequently raising ambiguity in determining whether a voxel is void within a damaged volume. In such instances, appropriate computer vision methods based on image processing or DL are required to perform effective spatial denoising.

The angle of incidence is directly related to the scanning field of the structural damage. The previous study [15] has shown that when the incidence angle is oblique, it causes a restricted visual field of the damage area due to occlusion, resulting in loss of information during inspection. Therefore, when it is impossible to conduct perpendicular scanning, it becomes necessary to perform scanning from different viewpoints to acquire sufficient damage spatial information. Then, the damage space can be reconstructed or potentially predicted through various technologies like neural radiance fields (NeRF) or 3D shape completion. This procedure can be carried out before transmission with point clouds or after transmission using the generated binary matrices through the proposed method.

Although the damage size and depth may meet specific criteria for a high-quality point cloud description, the edge effect still exists. As defined in [15], this effect refers to using an average object distance when the laser beam covers multiple surfaces with varying depths along the edge. It will blur the damage boundary in the point cloud and can only be mitigated by reducing the size of the laser beam. Therefore, a modest enhancement of the points' depth along the edge would benefit damage segmentation in the pseudo grayscale images (i.e., equivalent to image sharpening) while also expanding the margin for reliable damage assessment.

6.11 Summary

This study presents a highly efficient framework for damage volumetric assessment and DT synchronisation based on the point cloud. It includes surface damage detection, damage spatial segmentation, and DT model synchronisation. The surface damage detection is achieved through a pre-trained DeepLabV3+ model on the pseudo grayscale images derived from the point cloud depth information. It avoids the drawbacks of using image and point cloud fusion, such as extra photo acquisition, complex coordinate transformation and potential errors. The approach is validated on the specimens created in the lab with different crack widths and depths. It demonstrates that the trained model from the real grayscale images for damage detection can be applied to the pseudo grayscale images when the damage satisfies certain conditions, such as appropriate width and normalisation threshold. Meanwhile, the natural damage and manual concave patterns can be distinguished via the DeepLabV3+ model by training on the annotated damage and non-damage images.

After surface damage detection, the bounding box and mask are utilised for damage spatial segmentation to remove the redundant point cloud. Then, the separated point cloud can be converted into a highly lightweight 3D binary matrix representing the spatial damage geometry through voxelization and binarization. Compared with the full-scale object's point cloud transmission, communication complexity can be significantly decreased (over 99%) by only transmitting the segmented target point cloud in a binary matrix. The binary matrix can be further compressed through lossless RLE (with a CR of over 50%) for efficient data transmission and practical downstream tasks. This approach addresses the communication complexity challenge when DT synchronises with voluminous point cloud data in 3D scanning surveys.

Finally, the transmitted 3D binary matrix and the geo-referencing coordinates can be utilised for different downstream tasks, including SHM and reality modelling. For example, the damage volumetric assessment can be easily achieved by summing up the elements of the binary matrix (as void voxel – 1 and occupied voxel – 0). This voxel-based method doesn't require elaborate manual cutting for each component like the previous method based on the convex-hull fitting [4]. It can perform well for real-world damage with a complicated non-convex geometry under an appropriate resolution. Moreover, the FE model can be updated geometrically through the binary matrix with lower time and space complexity (see Table 11). The BIM model can also be updated automatically as an IFC file using appropriate patterns (such as ellipses and bounding boxes) to fit the damage contour in each layer.

The intact framework is validated based on two case studies, i.e., a synthetic groove and a real-world building crack. A comparative analysis, as shown in Table 2, demonstrates that the proposed framework is superior to the existing practices for damage assessment and reality modelling via 3D scanning.

Table 11 Comparative analysis between proposed methodology and existing practices

Methodology (data form)	**Payload**	**Volumetric Assessment**	**FE complexity (time/space)**	**BIM updating**
Project-based (full-scale PCD)	Heavy	Manually comparing	$O(P \times N)$ $O(M)$	Time-consuming
Research [4] (segmented PCD)	Medium	Manual cutting (convex)	$O(3 \times n \times log_2 n)$ $O(m)$	n/a
Proposed framework (binary matrix)	Light	Automatically	$O(n) + O(m)$ $O(m)$	Fast

Note: PCD – point cloud data; the damage volumetric estimation in the current project-based survey is achieved by manually comparing the full-scale point cloud with an ideal 3D model; P is the number of all the points in PCD; N is the number of elements, and M is the number of damaged elements in the full-scale model; n is the number of elements and m is the number of damaged elements in the target section.

It is important to highlight that the proposed framework and approaches hold promise for further study in extending their application to spatial damage on the object with a curved surface, such as cracks or spalling on a pillar. The proposed methodology can revolutionise the existing workflow for infrastructure maintenance surveys based on 3D scanning, which is unidirectional, time-consuming, and burdensome due to the large volume of point cloud data. Instead, it can facilitate near real-time damage assessment and reality modelling during the scanning process and provide timely feedback to the PE, effectively streaming the challenging 3D scanning survey into the infrastructure's DT.

Chapter 7 Time-series IFD and health monitoring

Bridge SHM and component fault diagnosis is essential for preventative maintenance (PM) in Industry 4.0. Data-driven approaches have been widely accepted for intelligent fault diagnosis (IFD) in smart manufacturing and infrastructure health-state monitoring, and various DL models have been developed for different datasets and scenarios.

However, an automatic and unified DL framework for developing IFD applications is still required. Hence, this chapter proposes an efficient framework integrating popular convolutional neural networks (CNNs) for IFD based on time-series data by leveraging AutoML and image-like data fusion at first. After normalisation, uniaxial or triaxial signals are reconstructed into 3-channel pseudo-images to satisfy the input requirements for CNNs and achieve data-level fusion simultaneously. Then, the model training, hyperparameter optimisation, and evaluation can be taken automatically based on AutoML. Finally, the selected model can be deployed on a cloud server or an edge device (via tinyML). The proposed framework and method are validated via two case studies, demonstrating the framework’s availability for automatic development of IFD applications and the effectiveness of the proposed data-level fusion method.

Secondly, an approach for bridge health-state recognition based on time-series signals from multiple sensors is proposed by using a multi-channel 1D-CNN architecture to achieve data-level fusion. The approach is validated via a public dataset from the VBM project of a railway bridge. Moreover, multiple sensors' data-level and decision-level fusion performances are compared and analysed.

7.1 IFD based on a uniaxial or triaxial sensor

7.1.1 Proposed IFD via AutoML-CNN and image-like fusion

7.1.1.1 Problem statement

As can be seen from the related works for IFD with DL, CNN-based pattern recognition using the derived 2D images from time-series data has become one of the most effective approaches for data-driven fault diagnosis. It can be attributed to the excellent feature learning capability of CNNs and subsequent FCNs fitting ability. Meanwhile, there are already many classical CNN architectures designed in computer vision, including LeNet, VGG, ResNet, EfficientNet, MobileNet, etc., as well as techniques developed for improvement, such as dilated convolution, attention, and lightweight design.

However, previous research has focused on implementing or improving an individual architecture, such as modified LeNet, VGG16, and transformer. Still, it did not involve different neural networks in a unified framework by leveraging AutoML. As is known, variant neural networks could perform differently in data-driven fault diagnosis even for the same dataset. Therefore, how to automatically realise training (including parameter optimisation) and select the most appropriate neural network has become an issue for developing practical IFD applications. Meanwhile, how to fuse the data from a triaxial sensor, such as 3-axis acceleration on (x, y, z), efficiently and effectively is also a problem.

7.1.1.2 Pseudo-image reconstruction and data fusion

The previous time-frequency transformation from 1D time-series signals to 2D synthetic images is usually time-consuming (e.g., the wavelet transformation for a sliding window of 1032 will take 1.653s on Google Colab) and requires expert knowledge of the frequency spectrum. In contrast, the spatial reconstruction from the same time-series sliding window to a grayscale image like in [176] will only take 0.0001s. However, the generated single-channel grayscale image cannot be utilised directly as input for the popular deep CNNs because they are designed for the 3-channel RGB images. Hence, an improved 3-channel pseudo-image reconstruction (i.e., imaging) method is proposed here, as shown in Figure 70.

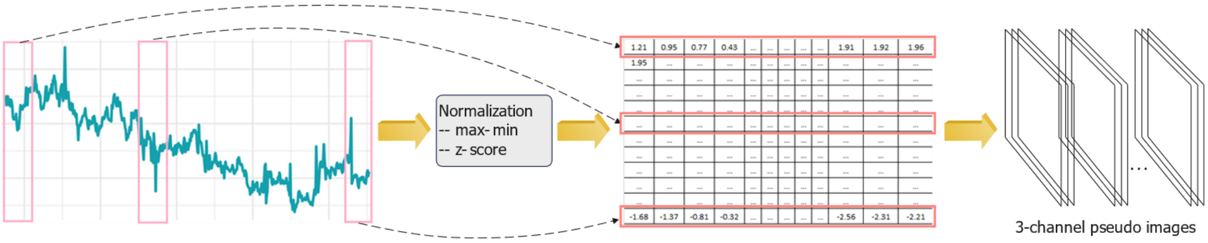


Figure 69 Proposed 3-channel pseudo-image reconstruction from time series

The first step in pre-processing is to select an appropriate sliding window size, which depends on the sampling frequency, computing capability (for edge device), etc. Data normalisation is suggested to decrease the time cost of training convergence, which can be the min-max normalisation or z-score standardisation. The pseudo-image pixels (i.e., matrix element) can be decimals without scaling up to the range of [0,255] (i.e., unlike Eq. 1 in previous research) because neural networks can convert the decimals to the scores between [0,1] after the hidden layers and softmax functions. The slice of signals on each axis is reshaped as a single-channel pseudo-image in rows or columns. Then, the single-channel pseudo-image from a uniaxial signal can be duplicated to 3 channels, and the slice of triaxial signals can be reconstructed into a 3-channel pseudo-image by stacking the single-channel image from each axis. The latter can achieve triaxial data-level fusion and satisfy the input requirement for CNN architectures at the same time.

7.1.1.3 Automated machine learning

AutoML includes the end-to-end procedure from beginning with a raw dataset to building a ML model ready for deployment. The high degree of automation in AutoML aims to allow non-experts to use ML models and techniques without requiring them to become experts in ML [255]. Currently, most popular CNN architectures have already been built as APIs in the mainstream DL framework, including Keras, TensorFlow, PyTorch, etc. They can be revoked straightforwardly, which serves as the foundation of AutoML in this study.

After the proposed imaging, the derived 3-channel pseudo-images are adopted as the input for integrated DL neural networks, which can be the built-in classical CNN architectures or the self-defined models. It is worth noting that the integrated neural networks are not limited to CNNs and can be any DNN architecture designed for RGB images, such as the Swim Transformer. The pseudo-images need to be resized appropriately according to the input requirement of each neural network. Then, the AutoML procedure can be carried out as shown in Figure 71, consisting of 1) automatic training through the popular DL frameworks for the integrated CNN architectures; 2) neural network search (and hyperparameter optimisation) based on evaluation according to various metrics; and 3) deployment on an edge device through tinyML.

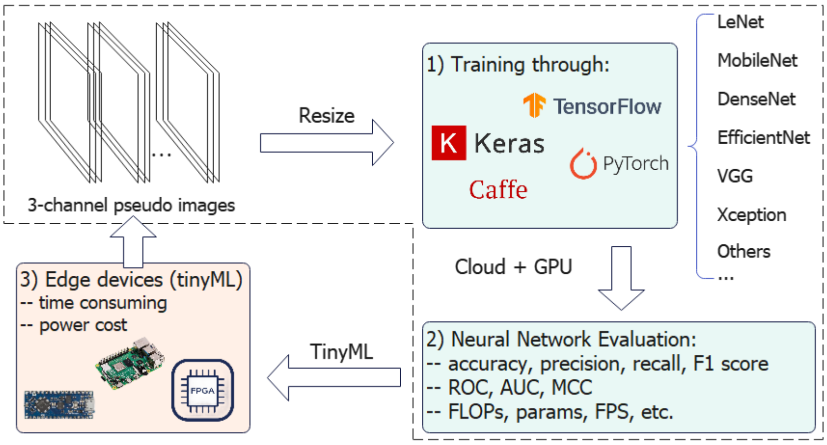


Figure 70 Proposed AutoML procedures for IFD

Notably, the first two steps should be taken on a high-performance computer, such as a cloud server with a GPU, because DL training requires considerable computing power and memory. Hyperparameters, including optimiser, epoch, activation function, and learning rate, are also available for automatic optimisation via different approaches, such as random search, grid search, Hyperband [256], Bayesian hyperparameter optimisation (BHO) [257], tree-structured Parzen estimator (TPE) [258], population-based training (PBT) [259]. Appropriate transfer learning, such as pre-trained backbones from similar signals, can also be integrated into the training step, especially when applying self-defined neural networks.

The models are evaluated via different metrics (see Eq. 40 - 45), including accuracy, precision, recall, F1 score, receiver operating characteristic curve (ROC), area under the ROC curve (AUC), Matthew’s correlation coefficient (MCC), etc.

$$\text{Accuracy } = \frac{TN + TP}{TN + FP + TP + FN} \tag{40}$$

$$\text{Precision } = \frac{TP}{TP + FP} \tag{41}$$

$$\text{Recall } = \frac{TP}{TP + FN} \tag{42}$$

$$F1 \text{ Score } = 2 \times \frac{\text{Precision } \times \text{ Recall}}{\text{Precision } + \text{ Recall}} \tag{43}$$

$$\text{MCC} = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{44}$$

Where TP – true positive, TN – true negative, FP – false positive, and FN – false negative.

$$AUC = \frac{\sum(p_i, n_j)_{p_i > n_j}}{P * N} \tag{45}$$

Where P – the number of positive examples, N – the number of negative examples, p_i – the prediction score for a positive example, and n_j – the prediction score for a negative example.

Additionally, because the float point operations (FLOPs) represent the forward-pass computing capability needed by the neural network model, the number of model parameters (params) is subject to the computing memory, and the frame per second (FPS) reflects the processing speed, if the trained models have similar performance using the above indicators, the one with fewer FLOPs, fewer params, and higher FPS would be recommended for practical applications.

Finally, the selected DL model can be deployed on edge devices for IFD by leveraging tinyML, such as TensorFlow Lite. Moreover, as edge devices are also usually the equipment for data acquisition or aggregation, the newly collected data can be used to update the training set based on supervised or semi-supervised learning via appropriate annotation, thereby enhancing the long-term performance of the IFD application, shown as the loop in Figure 71.

7.1.1.4 Proposed IFD framework and workflow

The complete workflow for IFD by leveraging AutoML-CNN and image-like data fusion can be seen in Figure 72. The time-series signals from uniaxial and triaxial sensors are adopted as the input for the built-in and self-defined CNN architectures seamlessly after the proposed pseudo-image reconstruction, achieving triaxial data fusion simultaneously. Neural network selection and hyperparameter optimisation can be implemented through AutoML based on model evaluation according to different metrics, including test performance (such as accuracy, precision, recall, F1 score, ROC, AUC, and MCC) and computing performance (such as FLOPs, params, and FPS).

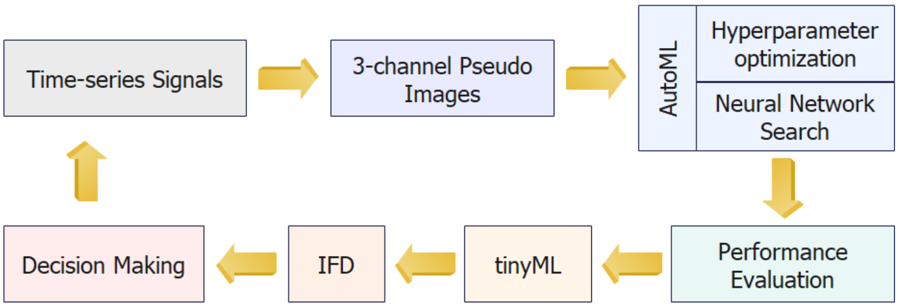
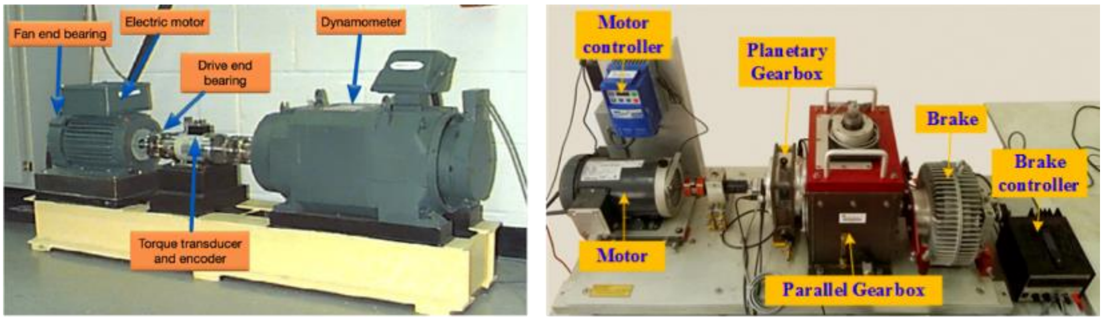


Figure 71 Proposed framework leveraging AutoML-CNN and image-like data fusion

7.1.2 Framework validation

7.1.2.1 Experiment preparation

The proposed framework, including the data fusion approach and the AutoML procedure for IFD, is validated via two case studies using the data from the CWRU and the SEU test rigs, as shown in Figure 73 a,b. The experiments are carried out on Google CoLab using a T4 GPU. The tf.keras provides model architectures, including the popular CNN architectures via the built-in APIs (such as Mobilenet, EfficientNet, Xception, and VGG16) and the self-defined classical models such as LeNet-5.



(a) (b)

Figure 72 CWRU bearing (a) and SEU gearbox (b) test rigs for data acquisition

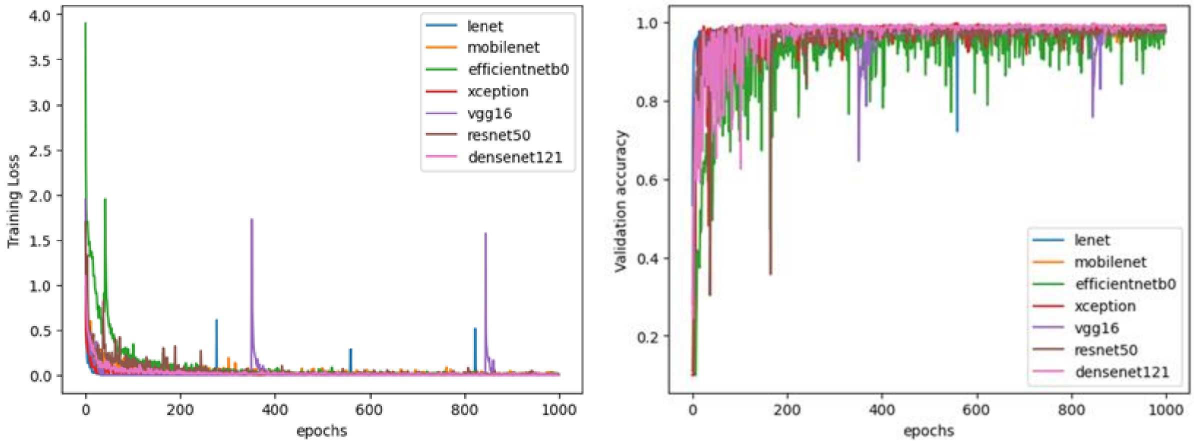
7.1.2.2 Case 1 – CWRU dataset (uniaxial signals)

In the first case, the bearing dataset collected by the Case Western Reserve University Bearing Data Center on a bearing test rig is utilised for framework validation with uniaxial signals [35]. The vibration signals in the experiment were collected from the uniaxial accelerometers on the drive end of the motor under one hp at the sampling frequency of 48kHz. Different faulty bearings were introduced with fault diameters of 0.007, 0.014, and 0.021 inches on the rolling element, the inner raceway, and the outer raceway, respectively. Therefore, there are nine fault categories plus a normal baseline, i.e., 10 kinds of bearing health states. The experiment aims to automatically recognise each fault category and select the most appropriate neural network for deployment through the proposed IFD framework.

Firstly, the uniaxial acceleration signals for each bearing health condition are separated into segments with the size of 1024 because the 32×32 pseudo-images can be utilised for most built-in APIs of classical CNN architectures directly in the tf.keras. The segments are split randomly into the training, validation, and test sets according to 60%:20%:20%, i.e., 2820, 940, and 940 segments, respectively. Z-score standardisation is employed on the training set, and the fitted scaler transforms the test set. The segments are reshaped to single-channel matrices and duplicated into triple-channel pseudo-images through the pipeline in Figure 70. Subsequently, the pseudo-images are provided to the integrated CNN architectures as input for training and evaluation. Here, the pseudo-images are resized to 75×75 through nearest-neighbour interpolation when necessary to meet the input shape requirements of some CNN architectures, such as Xception. A fixed training configuration is employed in the experiment to test the framework availability for neural network selection, as shown in Table 11. Automatic hyperparameter optimisation can be further integrated in future work. The training loss and test accuracy are shown in Figure 74.

Table 12 Training configuration

Input Shape	Split	Epochs	Optimiser	Batch Size	Learning Rate
32×32×3 or 75×75×3	60%:20%:20%	1000	Adam	32	0.001



(a) (b)

Figure 73 IFD experiment for uniaxial acceleration data via the framework: (a) training loss; (b) validation accuracy

The checkpoint with the highest validation accuracy during training is saved as the best model for each CNN architecture. Their test performance can be seen in Figure 75, including accuracy, precision, recall, F1 score, and normal-vs-fault AUC. As can be seen, the Xception model with resized pseudo-images (75×75×3) as input has the best performance, and its confusion matrix is shown in Figure 76. The FLOPs, parameters, and average FPS (within 100 times) are shown in Table 12. After conversion through TFLiteConverter [36], the derived lightweight Xception model can be deployed on an edge device, i.e., Raspberry Pi 4 (4GB) here, to satisfy the requirement for a practical application. It demonstrates that the proposed framework can achieve the model training, evaluation, and selection for IFD with the time-series signals from a uniaxial sensor by leveraging the popular built-in and self-defined CNN architectures based on AutoML, i.e., AutoML-CNN.

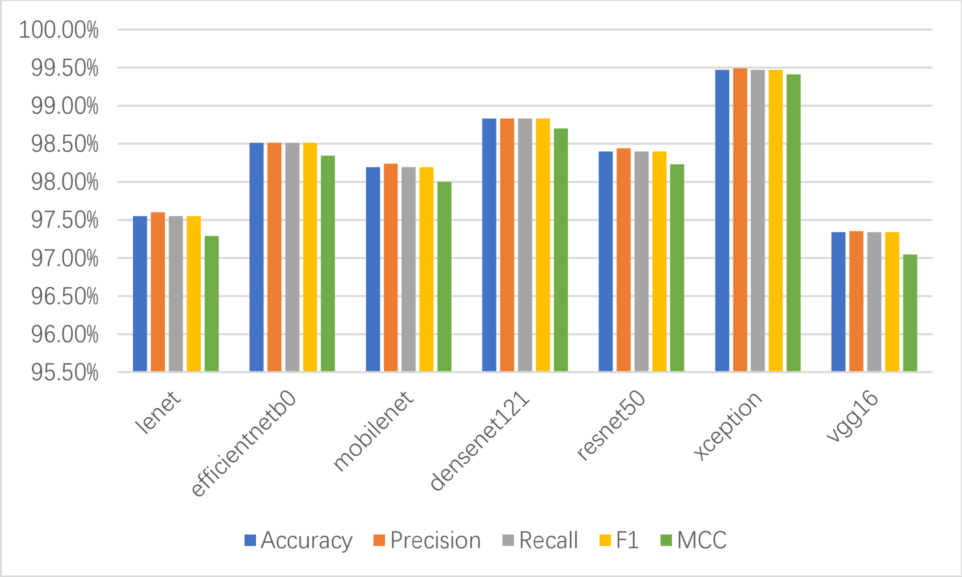
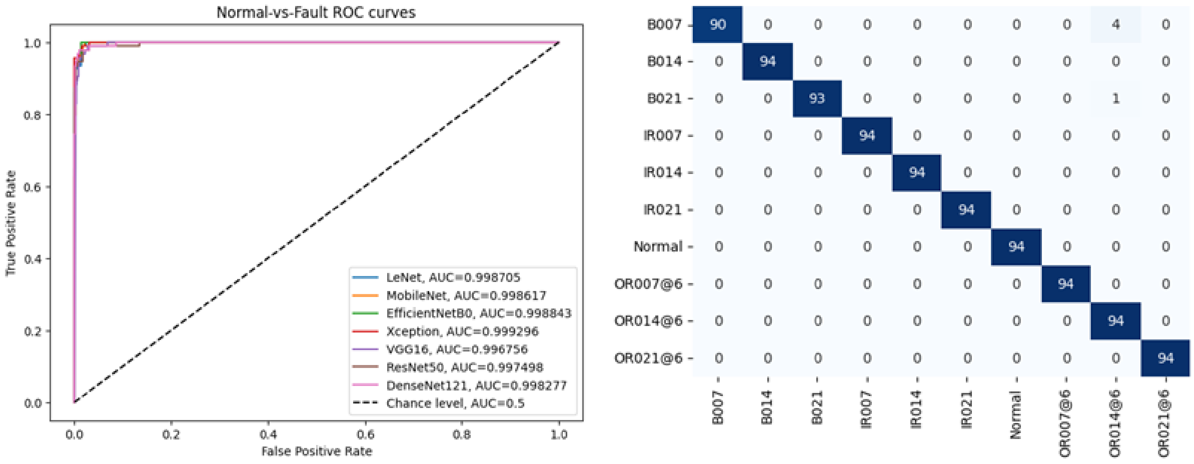


Figure 74 Test performance on the CWRU dataset through the proposed pipeline



	B007	B014	B021	IR007	IR014	IR021	Normal	OR007@6	OR014@6	OR021@6
B007	90	0	0	0	0	0	0	0	4	0
B014	0	94	0	0	0	0	0	0	0	0
B021	0	0	93	0	0	0	0	0	1	0
IR007	0	0	0	94	0	0	0	0	0	0
IR014	0	0	0	0	94	0	0	0	0	0
IR021	0	0	0	0	0	94	0	0	0	0
Normal	0	0	0	0	0	0	94	0	0	0
OR007@6	0	0	0	0	0	0	0	94	0	0
OR014@6	0	0	0	0	0	0	0	0	94	0
OR021@6	0	0	0	0	0	0	0	0	0	94

(a) (b)

Figure 75 (a) AUC for each model on the CWRU dataset and (b) confusion matrix of Xception on the CWRU dataset

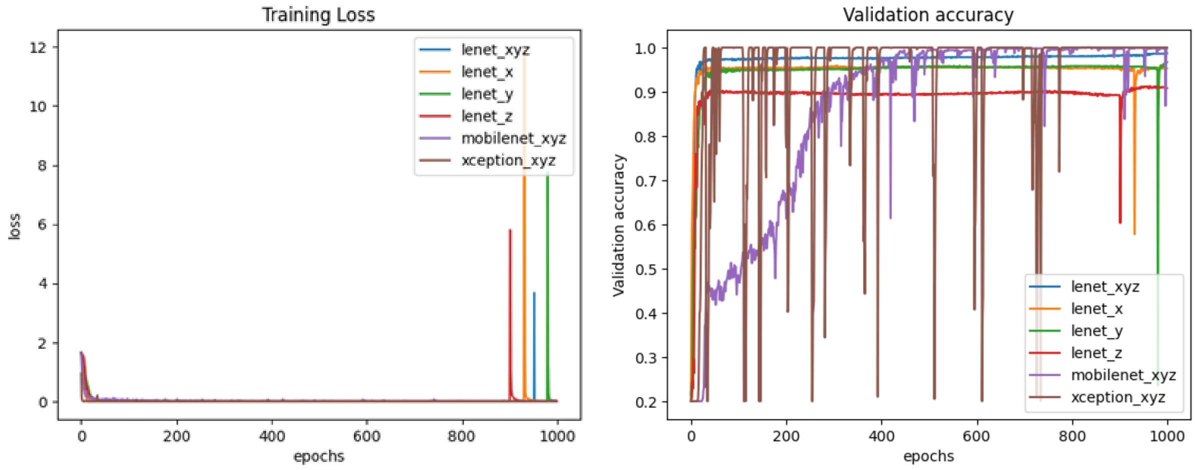
Table 13 CWRU model FLOPs, Parameters, and FPS

Models	LeNet	EfficientNetB0	Mobile-Net	Densnet-121	ResNet50	Xception	VGG16
FLOPs	6.58×10^5	8.66×10^6	1.16×10^7	5.79×10^7	7.89×10^7	5.62×10^8	3.32×10^8
Params	6.16×10^4	4.06×10^6	3.23×10^6	7.05×10^6	2.36×10^7	2.09×10^7	3.36×10^7
FPS	5449	2374	4058	1464	2463	1128	2760

7.1.2.3 Case 2 – SEU dataset (triaxial signals)

In the second case, the gearbox dataset collected on the DDS (Drivetrain Dynamic Simulator) test rig of Southeast University is utilised for framework validation with triaxial signals. The planetary vibration data on triple axes (i.e., x, y, z) under the load configuration 30-2 is adopted for the experiment. There are four gear faults, including chipped tooth, missing tooth, root fault, surface fault, plus health working state, i.e., five kinds of gear health states. The experiment aims to automatically recognise each fault category and select the most appropriate neural network for deployment through the proposed IFD framework.

Initially, the planetary vibration signals for each axis are separated into segments with a size of 1024. Then, the segments are split randomly into the training, validation, and test sets under 60%:20%:20%, i.e., 3100, 1000, and 1000 segments, respectively. Z-score standardisation is employed on the training set, and the fitted scaler transforms the test set. Moreover, the segments are reconstructed into 3-channel pseudo-images by stacking the single-channel image from each axis to achieve triaxial data fusion. Subsequently, the pseudo-images are provided to the integrated CNN architectures as input for training and evaluation. Here, the pseudo-images are resized to 75×75 through nearest-neighbour interpolation to meet the input shape requirements of some CNN architectures, such as Xception, when necessary. Like case 1, a fixed training configuration is employed in the experiment, as shown in Table 11. The training loss and test accuracy are shown in Figure 77, where lenet_x, lenet_y, and lenet_z denote the LeNet-5 performance based on the data on a single axis. In contrast, lenet_xyz, mobile_xyz, and xception_xyz represent the model performance based on the triaxial data through the proposed image-like data fusion.



(a) (b)

Figure 76 IFD experiment for triaxial acceleration data via the framework: (a) training loss; (b) test accuracy

The test performance of each model, including accuracy, precision, recall, F1 score, and normal-vs-fault AUC, is shown in Figure 78, where x, y, z, and xyz denote the models with single- or triaxial signals. As can be seen, the model with the triaxial signals through the proposed image-like data fusion can achieve better performance than the model with the uniaxial signals, i.e., lenet_xyz performs better than lenet_x, lenet_y, and lenet_z. The Xception model with resized pseudo-images (75×75×3) as input has the best performance, and its confusion matrix is shown in Figure 79. The FLOPs, parameters, and average FPS (within 100 times) are shown in Table 13. After conversion through the TFLiteConverter [36], the derived lightweight Xception model can be deployed on the Raspberry Pi for practical applications. It demonstrates that data fusion and model training for IFD with the triaxial signals can be achieved through the proposed framework by leveraging AutoML-CNN and the proposed image-like data fusion.

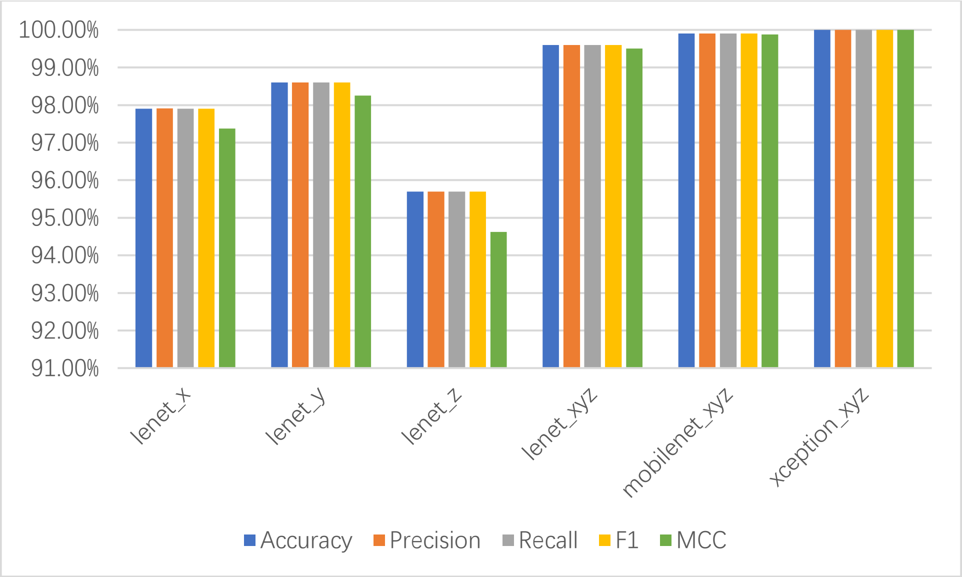
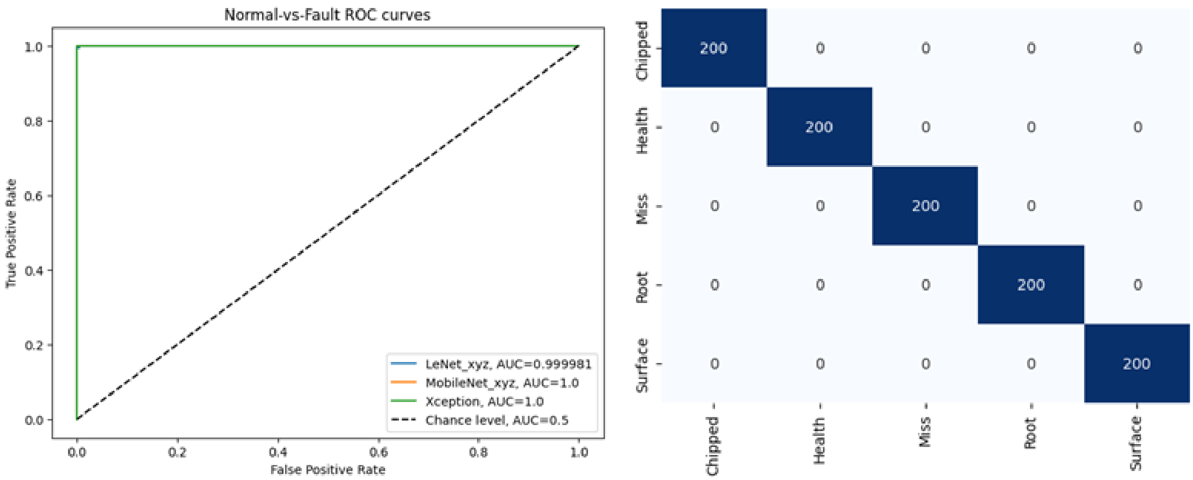


Figure 77 Test performance on the SEU dataset through the proposed pipeline



(a) (b)

Figure 78 (a) AUC for each model on the SEU dataset and (2) confusion matrix of Xception on the SEU dataset

Table 14 SEU model FLOPs, Parameters, and FPS

Models	LeNet_x	LeNet_y	LeNet_z	LeNet_xyz	Mobile-Net_xyz	Xception_xyz
FLOPs	6.58×10^5	8.66×10^6	6.58×10^5	6.58×10^5	6.58×10^5	5.62×10^8
Params	6.16×10^4	4.06×10^6	6.16×10^4	6.16×10^4	6.16×10^4	2.09×10^7
FPS	5778	5585	5726	6003	3493	1161

7.1.3 Summary

This work proposes an efficient and unified framework by leveraging AutoML and image-like data fusion for IFD with time-series signals from uniaxial or triaxial sensors. The popular built-in and self-defined DL architectures can be easily integrated into the framework to select the most suitable IFD model for different datasets or scenarios. Their training can be carried out consecutively or parallelly, and the evaluation can be taken automatically by comparing the model performance on the test set according to different metrics. In the proposed spatial reconstruction method, the time-series data from a uniaxial sensor can be reshaped into a 2D matrix after normalisation and then duplicated into a 3-channel pseudo-image. Similarly, the data from a triaxial sensor can be reconstructed into a 3-channel pseudo-image by stacking the single-channel image from each axis, thereby achieving data fusion.

The proposed IFD framework and the data fusion method are validated via two case studies based on uniaxial and triaxial vibration signals from the CWRU and SEU datasets, respectively. The experiments demonstrate that it can automatically achieve model training and evaluation through the proposed IFD framework, thereby enhancing the development efficiency for practical applications. Moreover, the fused triaxial time-series data through the proposed image-like data fusion method can improve the model performance effectively. Moreover, the recommended DL model can be easily deployed on a cloud server or an edge device (such as Raspberry Pi) via tinyML for inference to satisfy the requirement by practical scenarios, such as in a DT system, which requires timely and resilient decision-making, even under communication-constraint circumstances.

Although the proposed framework can benefit practical IFD applications by leveraging AutoML and image-like data fusion, it still has some limitations. Firstly, the proposed data-level fusion method is only suitable for the signals from a single triaxial sensor or no more than three uniaxial sensors at the same sampling frequency. Hence, fusion methods for heterogeneous data from multiple sensors (more than three) with different sampling frequencies are required for future research. Secondly, as there is a trade-off between neural network performance and computing complexity, a more in-depth study for model recommendation considering practical scenarios, such as device computing capability, storage, and power, is also necessary for future work.

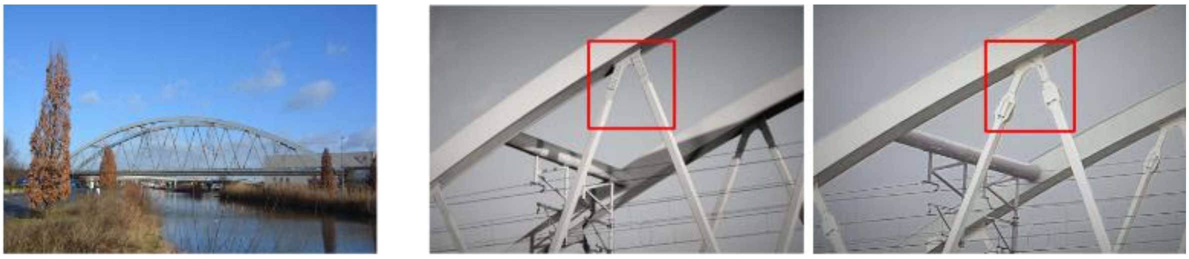
7.2 Health-state recognition based on multiple sensors

7.2.1 Proposed multi-channel 1D-CNN architecture

The proposed approach by leveraging AutoML-CNN and image-like data fusion in Section 7.1 still has some limits, e.g., the approach can only deal with the signals from a single-axis or a triple-axis sensor. To handle time series from multiple sensors (more than 3) for IoT-based bridge health monitoring, a multi-channel 1D-CNN architecture is proposed for data fusion and fault diagnosis, as shown in Figure 81.

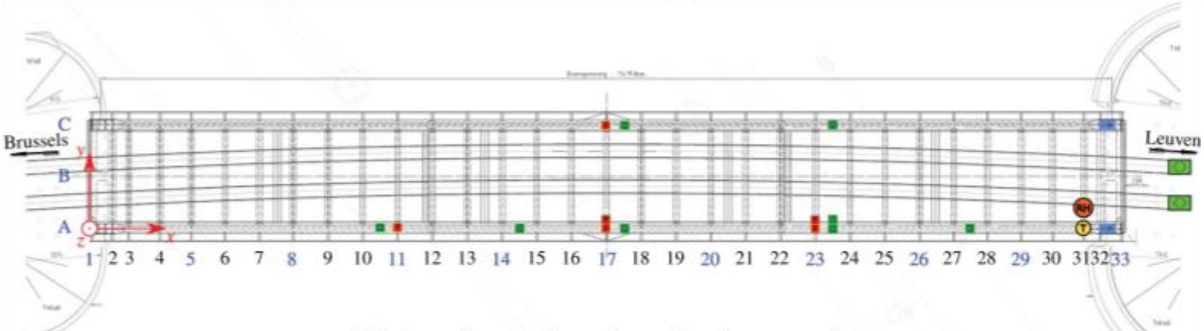
7.2.2 Experiment validation

A dataset of acceleration signals from the KW51 bridge VBM project [80] is employed here for approach validation, as shown in Figure 80. The multi-channel acceleration data are generated from 6 different uniaxial accelerometers triggered by train passes before and after a scheduled bridge repair (i.e., the damaged and healthy condition, respectively). Its sampling frequency and resolution are 825.8Hz and 24-bit, respectively. The time window in the experiment is 1 minute (i.e., time-series size of 50176 for each example). The data includes 361 examples under the defective bridge condition and 211 under the healthy bridge condition. The sensors are mounted on the bridge deck, and the detailed locations can be found in [80]. The train-validation-test split is 60%:20%:20%, i.e., 342:115:115.



(a) KW51 railway bridge

(b) Damage -- diagonal disconnection



(c) Accelerator locations (red squares)

Figure 79 KW51 railway bridge, damage, and accelerator locations

The experiment aims to identify the bridge's damaged and healthy status through the proposed 1D-CNN architecture. It starts from the data-level fusion, i.e., integrating all acceleration signals into a multi-channel time sequence. Then, compare the classification results of each single sensor and multiple sensors (i.e., with and without multi-channel fusion).

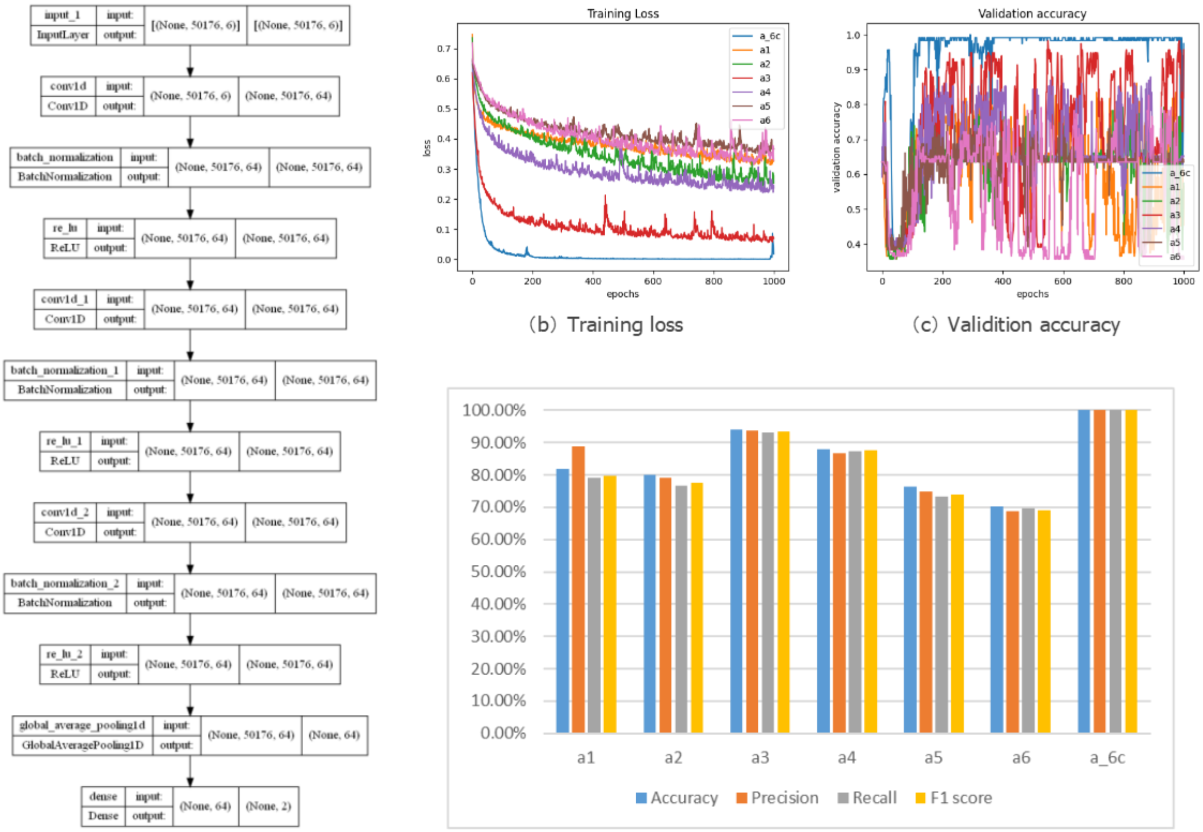
Moreover, the experiment also tests the decision-level fusion, i.e., ensemble voting for the results from each single sensor. The voting includes unweighted hard and soft voting. The former is based on majority or plurality voting for the results from multiple classifiers. In contrast, the latter is based on multiple classifiers' average predicted probability for each class. Then, both fusion methods are compared in test performance. The training configuration is presented in Table 14.

Table 15 Configuration for experiment with 1D-CNN

Input shape	Split	Epochs	Optimizer	Batch Size	Learning Rate
50176×1 or 50176×6	60%:20%:20%	1000	Adam	32	0.001

7.2.3 Result analysis

The proposed 1D-CNN architecture, training loss, validation accuracy, and test performance are shown in Figure 81 (b) (c) (d), where a1, a2 … a6 and a_6c denote each uniaxial acceleration and the combined 6-channel acceleration. As can be seen, the model with combined 6-channel acceleration can achieve much better accuracy (reaching 100%), precision, recall, and F1 score than the model with each uniaxial acceleration. It demonstrates the availability of the proposed 1D-CNN architecture for pattern recognition with time-series signals from multiple sensors by data-level fusion.



(b) Training loss

(c) Validition accuracy

(a) Proposed 1D-CNN

(d) Test perfromance

Figure 80 Proposed 1D-CNN and test performance at data-level fusion

Moreover, test performance via the decision-level fusion (including hard and soft voting) is shown in Figure 82, where hard_vote and soft_vote represent hard and soft voting results. As can be seen, unlike the previous research [179], the experiment does not see the enhancement for classification by decision-level fusion. Instead, it demonstrates that the data-level fusion performs much better than the decision-level fusion in this case, using the proposed 1D-CNN architecture, and the decision-level fusion is even uncompetitive versus most stand-alone sensors without fusion. This phenomenon may be attributed to different datasets and 1D-CNN architectures, which require more in-depth study in the future.

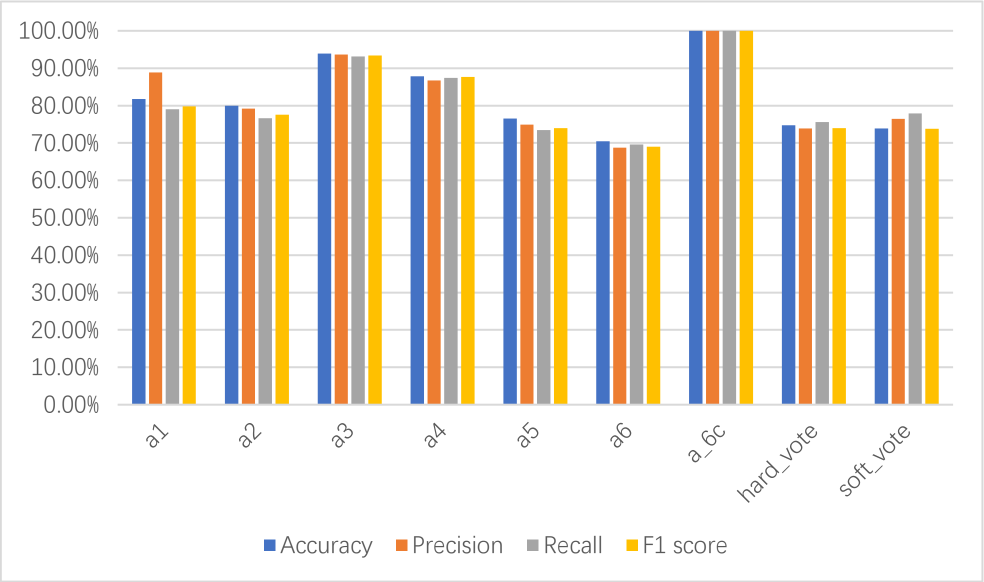


Figure 81 Comparison of decision-level and data-level fusion on the test set

Chapter 8 BMKG data mining with text encoding and GraphSAGE

KG is one of the most effective methods to document the domain knowledge of bridge maintenance. Many previous efforts have been made in ontology generation or KG completion, but few studies have focused on graph data mining after that. Hence, this chapter proposes a BMKG schema based on practical maintenance reports and explores graph data mining (including node classification and link prediction) on the BMKG by leveraging text encodings (through LLMs) and inductive GraphSAGE. The proposed approaches are validated on incomplete BMKGs with missing links. The models are trained from graph context to identify node layer information for semantic enrichment and provide repair suggestions for unsolved defects from the existing options. Finally, an intact workflow integrating the proposed KG schema and approaches is designed for routine practice of bridge maintenance.

8.1 Theoretical foundation

8.1.1 Text encoding

Various static word embedding techniques, such as Bag of Words [260], TF-IDF [261], Word2Vec [262] and GloVe [263], can be utilized to transform a single word to an n-dimensional vector by looking up through the created “tables”. The generated vectors are supposed to maintain attributes of the words from the training set and can be adopted for different downstream tasks, such as lexical similarity measurement. However, word embedding approaches encounter a few issues in generating the vectorized representation of a sentence, e.g., how to manipulate multiple embedding vectors. After BERT [264] was born, many pre-trained text models emerged based on large-scale unsupervised learning, which can be utilized for text encoding at the sentence level. Because the unsupervised learning tasks for training mainly involve mask language modelling (MLM), next sentence prediction, etc., they have excellent performance [265] in close tests, text comparison, entity extraction, and many other downstream tasks. These pre-trained large language models (LLM) can be divided into five categories according to their architectures, shown in Table 15.

Table 16 Pre-trained language models for text encoding [266]

Autoregressive	**Autoencoding**	**Seq2Seq**	**Multimodal**	**Retrieval-based**
GPT GPT-2 CTRL Transformer-XL Reformer XLNet	BERT ALBERT RoBERTa DistilBERT ConvBERT XLM XLM-RoBERTa FlauBERT ELECTRA Funnel Transformer Longformer	BART Pegasus MarianMT T5 MT5 Mbart ProphetNet XLM-ProphetNet	MMBT CLIP	DPR RAG

In this work, the graph nodes are based on text data from a structured bridge maintenance report. The text encoding through the pre-trained LLM aims to provide node features in the GNN for multiple downstream tasks. The complete encoding pipeline is illustrated using BERT with a series of text messages or sentences from the report, as shown in Figure 82.

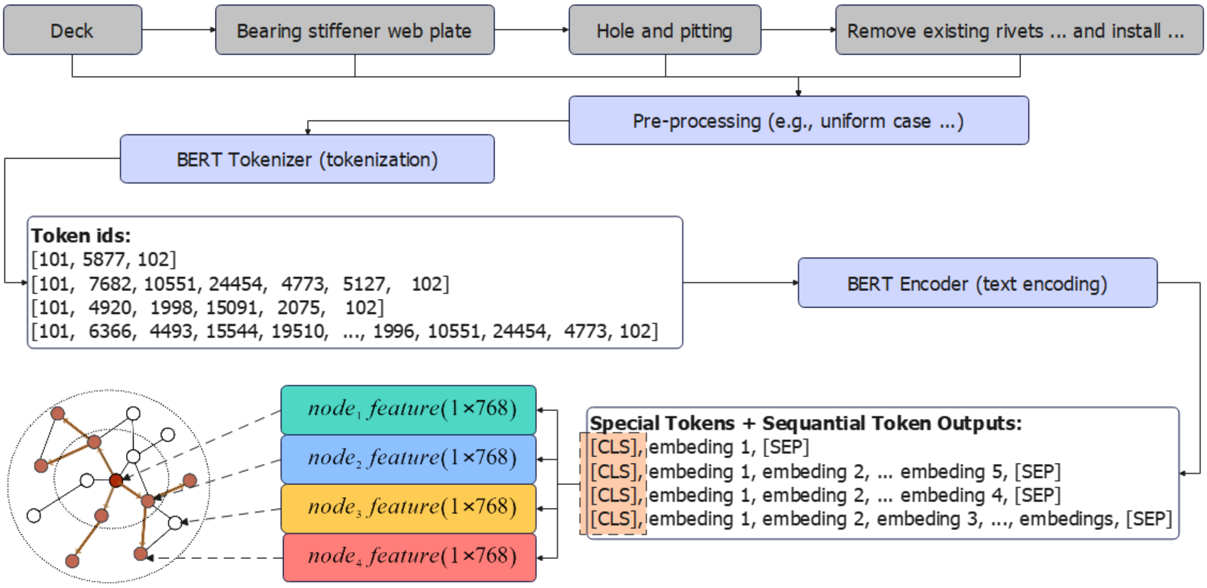

Deck
Bearing stiffener web plate
Hole and pitting
Remove existing rivets ... and install ...
Pre-processing (e.g., uniform case ...)
BERT Tokenizer (tokenization)
Token ids:
[101, 5877, 102]
[101, 7682, 10551, 24454, 4773, 5127, 102]
[101, 4920, 1998, 15091, 2075, 102]
[101, 6366, 4493, 15544, 19510, ..., 1996, 10551, 24454, 4773, 102]
BERT Encoder (text encoding)
Special Tokens + Sequantial Token Outputs:
[CLS], embeding 1, [SEP]
[CLS], embeding 1, embeding 2, ... embeding 5, [SEP]
[CLS], embeding 1, embeding 2, ... embeding 4, [SEP]
[CLS], embeding 1, embeding 2, embeding 3, ..., embedings, [SEP]
$node_1\ feature(1\times768)$
$node_2\ feature(1\times768)$
$node_3\ feature(1\times768)$
$node_4\ feature(1\times768)$

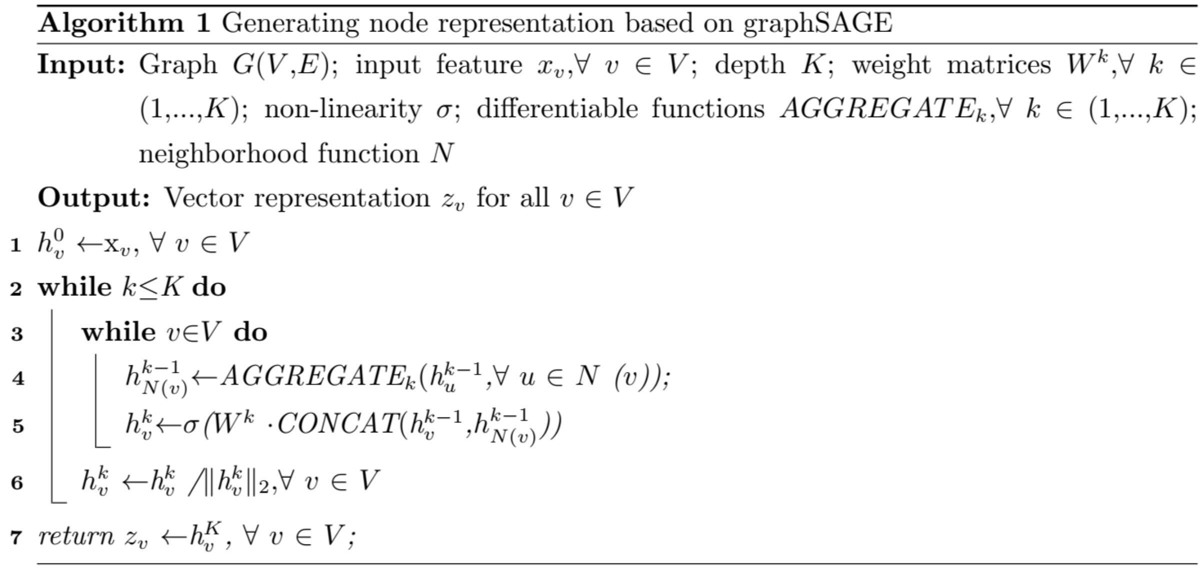
Figure 82 Text encoding with BERT to generate node features

Firstly, each node message or sentence in BMKG is separated into multiple tokens in the id form, whilst the special tokens [CLS] and [SEP] (i.e., the ids of 101 and 102) are added at the head and end, respectively. After encoding through the pre-trained text encoder, the hidden layer output of the special token CLS (used for classification), which has been proven as the effective sentence-level representation [264], [267], is adopted as the latent feature of the node in the graph. These CLS embedding vectors have identical dimensions (i.e., 768 here), which makes feature input easy in GNN.

8.1.2 GraphSAGE

Most GNNs are inherently transductive (such as GCN) and can only generate representation through all the nodes for a single fixed graph [268], i.e., the node representation learned from transductive approaches uniquely corresponds to the graph. Such graph representation frameworks are difficult to generalize to unseen nodes in an evolving graph with new links. Meanwhile, they cannot learn to generalize across different graphs. In contrast, GraphSAGE is an inductive framework to generate low-dimensional vector representations for nodes, which is especially useful for graphs with rich node attribute information [269]. It aims to solve the above issues by learning a dynamic representation method across partial graphs through inductive training. After training, GraphSAGE can be used to generate node representation for previously unseen nodes or entirely new input graphs if these graphs have the same attribute schema as the training data. Meanwhile, the learned node representation can change along with the neighbour relationship variation in an evolving graph. Therefore, GraphSAGE is particularly suitable for link prediction in an incomplete graph with missing links.

Assuming there are K aggregator functions, denoted as $AGGREGATE_k$ ($\forall k \in \{1, \dots, K\}$), which are used for aggregating features from node neighbours. Their set of weight matrices is denoted as $\boldsymbol{W}^k$ ($\forall k \in \{1, \dots, K\}$), which are utilized for message propagation across different layers of the model [269]. The algorithm to generate node vector representation based on GraphSAGE can be described using the pseudo-code below.



Algorithm 1 Generating node representation based on graphSAGE

```
Input: Graph $G(V,E)$; input feature $x_v, \forall\, v \in V$; depth $K$; weight matrices $W^k, \forall\, k \in (1,...,K)$; non-linearity $\sigma$; differentiable functions $AGGREGATE_k, \forall\, k \in (1,...,K)$; neighborhood function $N$
Output: Vector representation $z_v$ for all $v \in V$
$h_v^0 \leftarrow \mathrm{x}_v, \forall\, v \in V$
while $k \leq K$ do
    while $v \in V$ do
        $h_{N(v)}^{k-1} \leftarrow AGGREGATE_k(h_u^{k-1}, \forall\, u \in N\ (v));$
        $h_v^k \leftarrow \sigma(W^k \cdot CONCAT(h_v^{k-1}, h_{N(v)}^{k-1}))$
    $h_v^k \leftarrow h_v^k /\|h_v^k\|_2, \forall\, v \in V$
return $z_v \leftarrow h_v^K, \forall\, v \in V;$
```

The principle behind the GraphSAGE embedding algorithm is that nodes can aggregate more and more information from their neighbours and reach further parts of the graph as this process iterates. In detail, k denotes the current step in the loop (or the depth of the search), and h^k denotes the latent embedding at step k (or the hidden layer k). At first, each node $v \in V$ aggregates its neighbourhood embeddings, i.e., h_u^{k-1} $\forall u \in N(v)$, into a single vector $h_{N(v)}^{k-1}$. After aggregation, GraphSAGE concatenate the node's embedding h_v^{k-1} with the aggregated neighbourhood embedding $h_{N(v)}^{k-1}$, and then fed the result into a fully connected neural network (FCN) with a nonlinear activating function σ to generate the representation vector (i.e., h_v^k) for the next step. It is worth noting $k-1$ indicates the aggregation, and the node's embedding employs the related node representation vectors at the previous step (not the current k). Here, the initial case, i.e., $k=0$, denotes the input node features. Finally, the representation output at the depth K is obtained after normalization as $z_v = h_v^K,\ \forall v \in V$.

In GraphSAGE, the aggregation of neighbour embeddings, i.e., $AGGREGATE_k$, can be done by various architectures [269], including mean, pool, GCN, and LSTM. Here, the mean aggregator used in this work is indicated in Eq. 46.

$$\boldsymbol{h}_v^k \leftarrow \sigma(\boldsymbol{W} \cdot MEAN(\{\boldsymbol{h}_v^{k-1}\} \cup \{\boldsymbol{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\}) \tag{46}$$

8.2 Proposed approaches for maintenance knowledge mining

8.2.1 Knowledge-graph schema

A maintenance-oriented knowledge-graph schema for bridges is proposed as shown in Figure 83, which consists of five layers, i.e., primary structure layer, component layer, element (or minor component) layer, defect (or fault) layer, and repair action layer. This knowledge graph schema is established based on observations from routine practice: 1) bridges always present symmetry and repeatability, e.g., different spans in a bridge usually have similar components and defects; 2) defect assessment and repair suggestions usually present semantic similarity and hierarchy, e.g., medium section loss (less than 50%) and remarkable section loss (more than 50%) for stiffener cranks have similar and incremental repair actions. Such semantic similarity can also be observed in historical textual reports and can be leveraged for maintenance recommendation. For example, similar defects on different components are promising to adopt the analogous repair proposals. Additionally, it is worth noting that the proposed BMKG schema is directed from top to bottom (i.e., from the primary structure layer to the action layer). All connections are strictly limited between the nodes in adjacent layers. This setting is critical for data mining in the BMKG with graph-based algorithms.

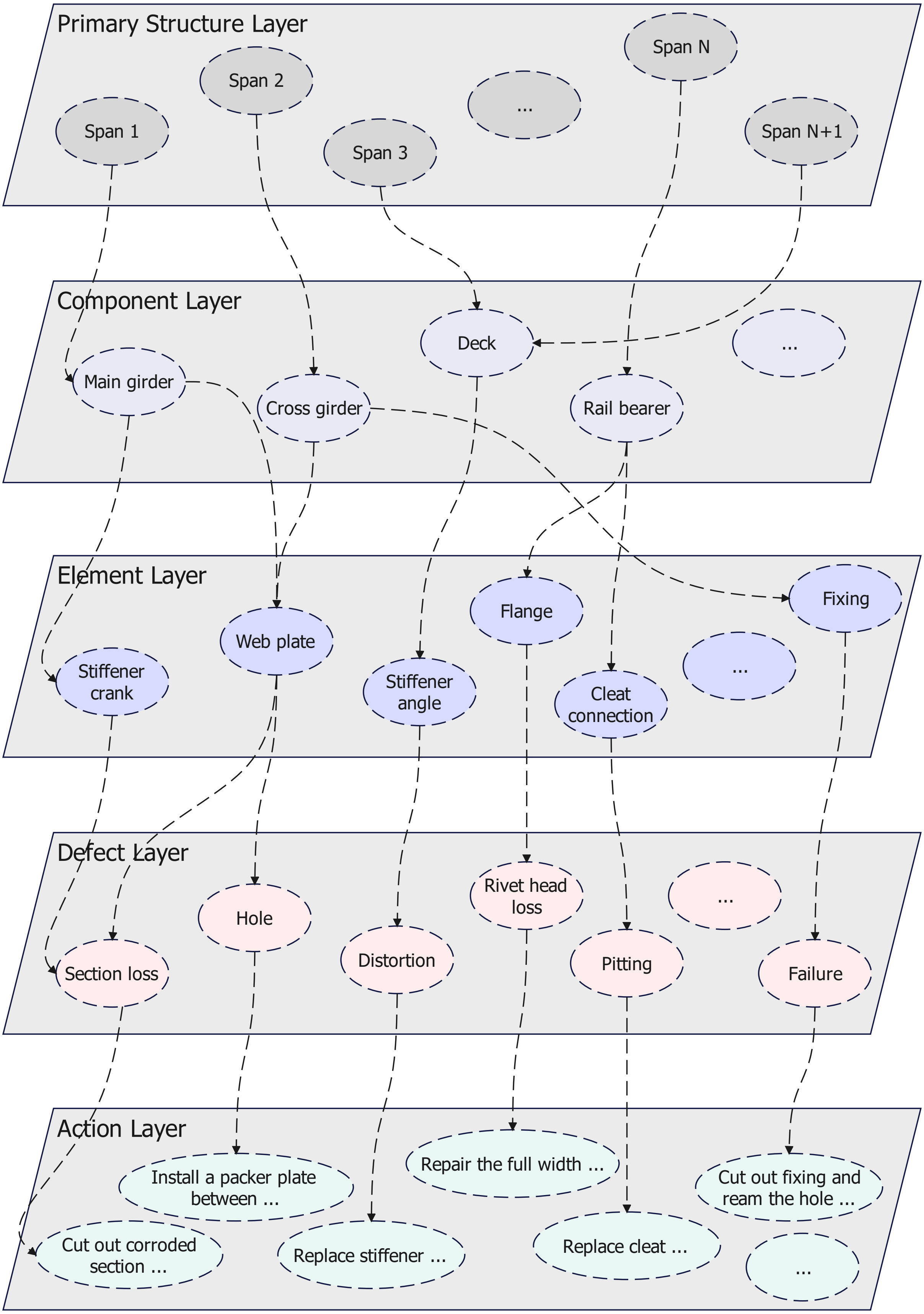
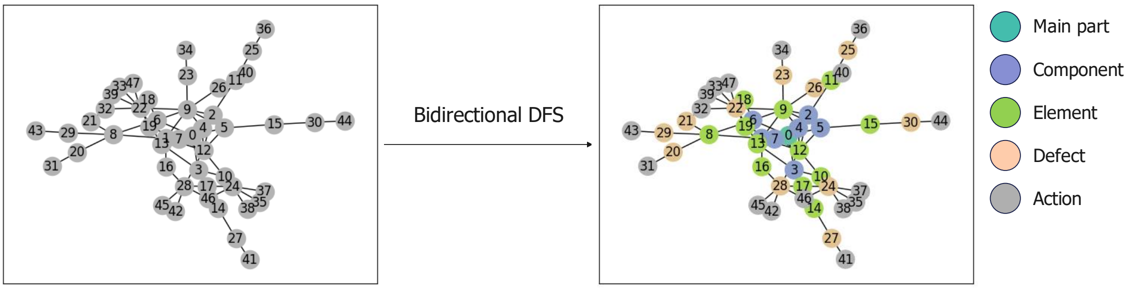


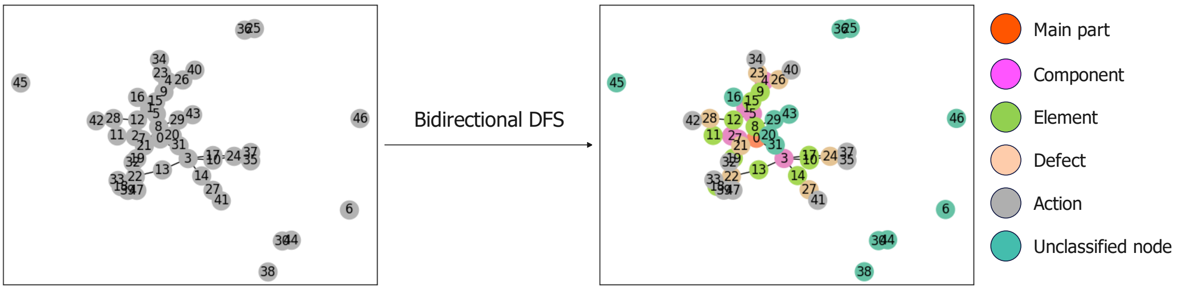
Figure 83 Proposed maintenance-oriented knowledge graph schema

8.2.2 Node classification

Node classification is a typical task for semantic enrichment of knowledge graphs in previous research [270], [271]. This task starts with depth-first search (DFS), a classical method to find the shortest path in a graph from one node to another. After indicating the staring nodes, i.e., the main part layer, it can be utilized to identify the following nodes' layer levels (i.e., node layer classification) in an intact BMKG graph, as shown in Figure 84 (a). Node layer information is helpful for different downstream tasks, such as link prediction, knowledge graph completion, and templated Q&A systems. In detail, one-way and bidirectional DFS can be successfully employed for node (level) classification in directed and undirected intact graphs, respectively. However, when the graph becomes incomplete with missing links, which usually happens due to data deficiency in node edges (e.g., missing entities or relations in the structured report), DFS will fail to classify all the node levels in the BMKG without any ancillary information, as shown in Figure 84 (b).



(a) Node layer classification using DFS for an intact graph



(b) Node layer classification using DFS for a graph with 30% missing links

Figure 84 Node layer classification using DFS on intact and incomplete graphs

Therefore, the encoded text embedding vectors are employed as node features to solve the above issue by leveraging the GNN. Here, the inductive GraphSAGE is utilized for node classification through supervised learning. The identified nodes after DFS are employed as the training set, while the unidentified nodes are adopted as the test set. The complete solution pipeline is proposed, as shown in Figure 85. Different pre-trained text encoders will be tested in the experiment for comparison to find the most appropriate encoder for the target dataset in bridge maintenance.

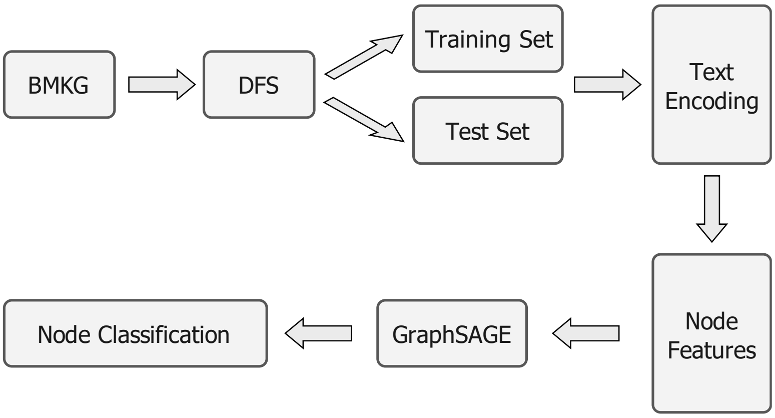


Figure 85 Proposed pipeline for node layer classification by leveraging DFS, text encoding, and GraphSAGE

8.2.3 Link prediction

In this work, maintenance suggestion is formulated as a link prediction problem, which predicts if an edge exists between two specific nodes, i.e., the missing links. The problem can be taken as binary classification, and contrastive learning can be utilized to solve this problem, following the steps listed below [272].

1) Firstly, the existing edges in the graph are treated as positive examples.
2) Then, a few non-existent edges (i.e., node pairs without edges) are sampled as negative examples.
3) The positive and the negative examples are separated into training and test sets.
4) Finally, the model is evaluated using a binary classification metric such as Area

Under Curve (AUC).

In this work, the pairs of nodes are treated as graphs, i.e., positive and negative graphs constructed with positive and negative examples, respectively. Meanwhile, they have the same set of nodes as the original graph, enabling node features to be utilised across different graphs, but noting that instead of generating node representation, link prediction needs to compute the node pair's (i.e., edge) representation, which is denoted as scalar pair-wise scores for inference. In practice, such pair-wise scores can be computed using a dot product of both node representations for each edge or a multilayer perceptron. Meanwhile, the positive and negative edge examples are {1, 1, …, 1} and {0, 0, …, 0}, respectively. After concatenation, training aims to minimize the difference between the generated pair-wise scores and the actual examples' binary labels, i.e., loss function. The pipeline for link prediction can be shown in Figure 86.

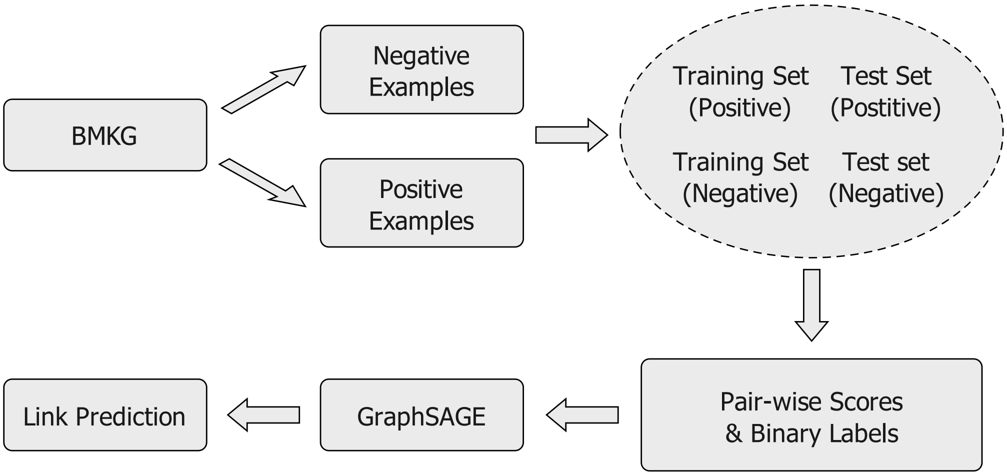


Figure 86 Proposed pipeline for link prediction by leveraging contrastive learning and GraphSAGE

8.2.4 Loss function and evaluation metrics

The loss functions for node classification and link prediction are based on multi-class cross-entropy and binary cross-entropy, respectively, which can be summarized in Eq. 47.

$$L = \frac{1}{N}\sum_{i} L_i = -\frac{1}{N}\sum_{i}\sum_{c=1}^{M} y_{ic}\, log_2(p_{ic}) \tag{47}$$

Where N is the number of examples; M is the number of classes and $c \in [1, M]$; y_{ic} is a binary value (0 or 1) – if the ith example's true class is c, $y_{ic} = 1$, and vice versa; p_{ic} is the probability of the ith example belonging to the class c, derived from the Softmax function (Eq. 48).

$$p_c = Softmax(h_j) = \frac{e^{h_j}}{\sum_{c=1}^{M} e^{h_c}} \tag{48}$$

Where p_c is the calculated probability of the example belonging to the class c; $j \in [1, M]$ indicates jth class; h_j is the jth node output of the last hidden layer (i.e., embeddings).

The evaluation metrics for node classification include accuracy, precision, recall, and f1 score, shown in Eq. 40 - 44.

The receiver operating characteristic (ROC) and the area under the ROC curve (i.e., AUC) are employed as the evaluation metrics for link prediction using contrastive learning. ROC is plotted in the x-y coordinate system, in which x represents the false positive rate (FPR) (Eq. 49), and y represents the true positive rate (TPR), i.e., recall.

$$FPR = \frac{FP}{FP + TN} \tag{49}$$

The AUC can be calculated as indicated in Eq. 50, which represents the probability that a positive example score is greater than a negative example score when they are both sampled out of data randomly.

$$AUC = \frac{\sum(p_i, n_j)_{p_i > n_j}}{P * N} \tag{50}$$

Where P – the number of positive examples; N – the number of negative examples; p_i – the prediction score for a positive example; n_j – the prediction score for a negative example.

8.3 Proof of concept

8.3.1 Experiment preparation

To validate the proposed approach, a BMKG is created based on a structured real-world bridge maintenance report (see Appendix I), including one primary structure part (span), seven components (main girder, cross girder, deck, rail bearer, caisson bracing …), twelve elements (stiffener crank, top flange, bottom flange, plate, web plate, gusset plate, fixing, cleat connection, …), eleven defects (excessive/medium/slight section loss, distortion, failure, hole and pitting, …), and seventeen repair proposals (cut out corroded section, weld a new plate, replace stiffener, install a bolted packer, remove the existing rivet, cut out fixing and ream the hole, …). The selected pre-trained language models for text encoding in the experiment include three autoregressive models (i.e., GPT2, Transformer-XL, Reformer), two autoencoding models (i.e., BERT, XLM), one Seq2Seq model (i.e., BART), and one multi-modal model (i.e., CLIP). The model details, including maximum input sequence length (Max_Seq), embedding dimension, and training corpus (or dataset), are shown in Table 16.

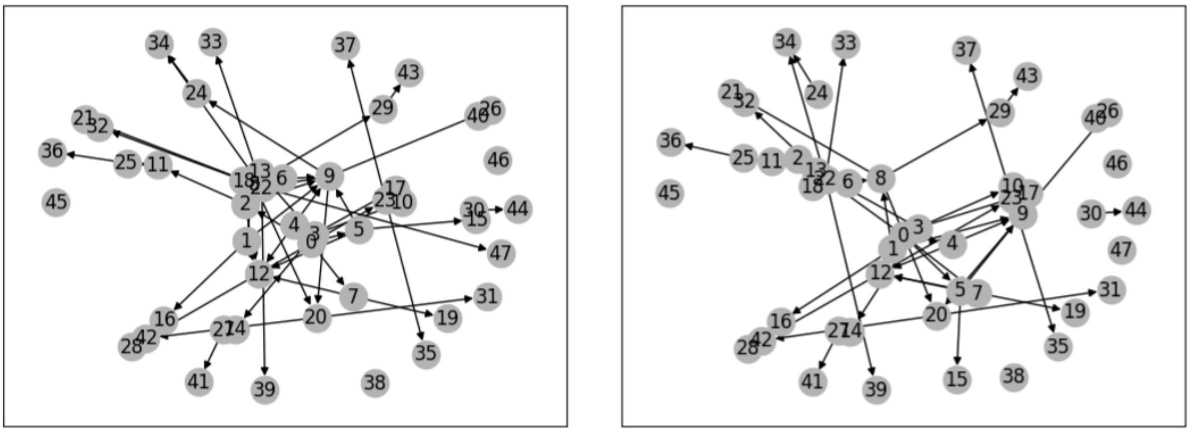
Table 17 Pre-trained language models for text encoding

Language Models	**Max_Seq**	**Dimension**	**Corpus / Dataset**
bert-base-uncased	512	768	Wikipedia + Book Corpus
bart-base	1024	768	GLUE and SQuAD
xlm-mlm-en-2048	512	2048	Wikipedia + Book Corpus
reformer-crime-and-punishment	4096	320	*Crime and Punishment*
gpt2	no limit	768	BookCorpus + 8 milion web pages
transfo-xl-wt103	no limit	1024	Wikitext-103

text encoder (CLIP)	77	512	400 million text and image pairs

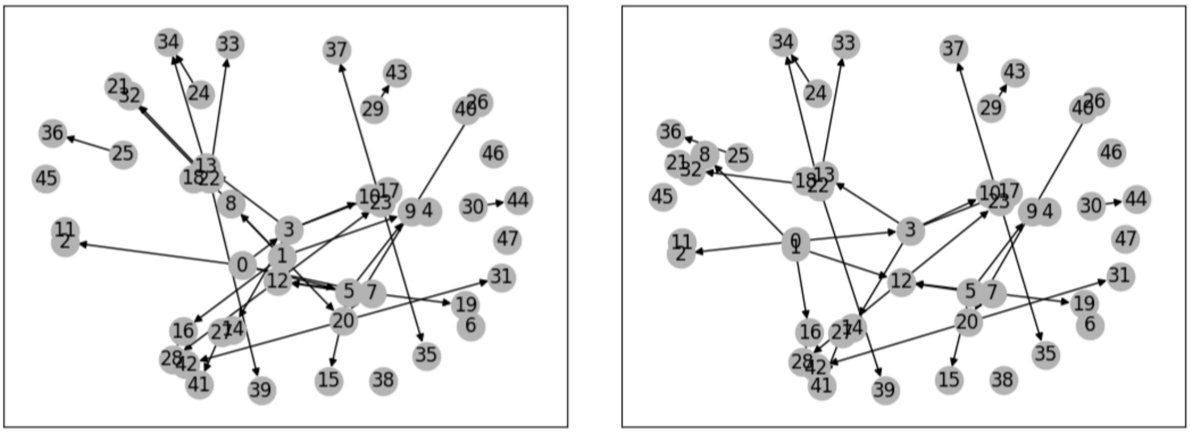
8.3.2 Node layer identification

The intact directional BMKG has 48 nodes and 66 edges, as shown in Figure 84 (a). All the nodes' layers of the intact BMKG can be identified through bidirectional DFS if the starting node 0 (in the primary structure layer) is known. The incomplete BMKGs in this study are obtained by randomly removing 10%, 20%, 30%, and 35% of links (i.e., edges). In contrast, incomplete BMKG after removing links will lead to isolated node clusters, which can be identified via DFS, as shown in Figure 87. This experiment aims to predict their layers in the BMKG via node classification using GNN and encoded text embeddings through LLMs (as node features).



BMKG with 10% missing links

BMKG with 20% missing links

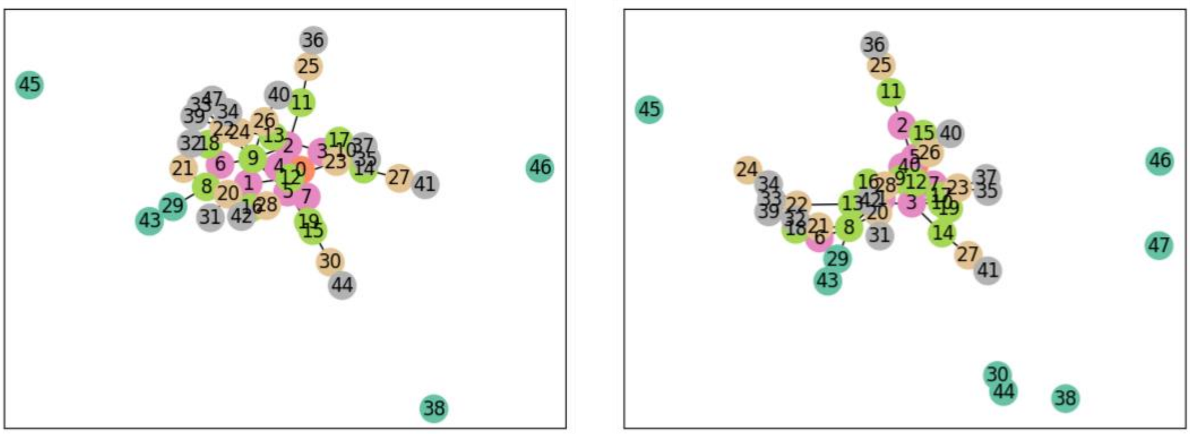


BMKG with 30% missing links

BMKG with 35% missing links

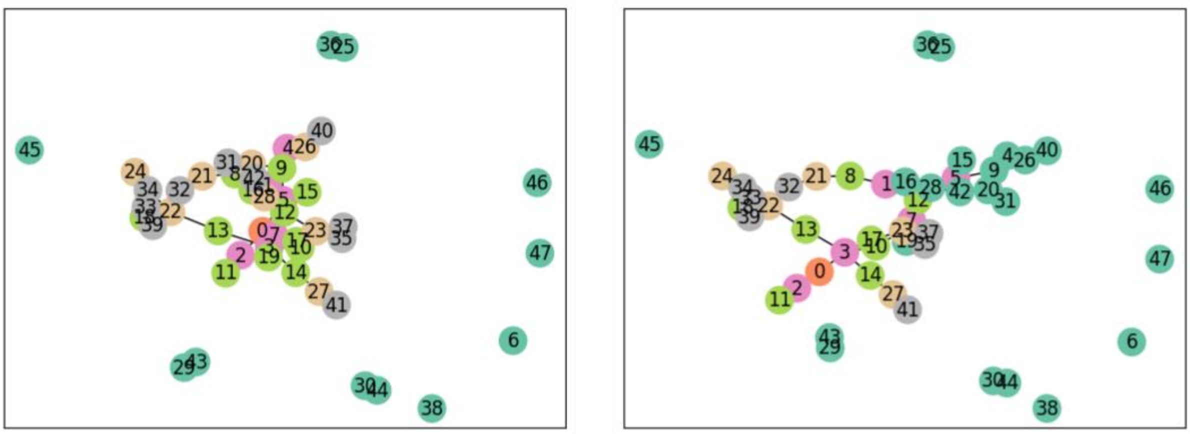
Figure 87 Incomplete BMKG with 10%-35% missing links

Firstly, bidirectional DFS is employed for node classification, and the results can be shown in Figure 88. Here, node 0 is in the first layer – the primary structure layer. Due to missing links, there are a few unclassified nodes after DFS, i.e., 6 missing links (10%) – 11 unclassified nodes; 13 missing links (20%) – 8 unclassified nodes; 19 missing links (20%) – 11 unclassified nodes; 23 missing links (35%) – 22 unclassified nodes.



10% missing links after DFS

20% missing links after DFS



30% missing links after DFS

35% missing links after DFS

● Main part ● Component ● Element ● Defect ● Action ● Unclassified

Figure 88 Incomplete BMKGs after bidirectional DFS

Then, the encoded text embeddings through different pre-trained language models (see Table.) for each node are adopted as node features to identify the node layer in the BMKG (i.e., node classification). The nodes with classified layers after DFS are taken as the training set, and the unclassified nodes are adopted as the test set. Root

Mean Squared Propagation (RMSprop) is utilized for hyperparameter optimization. The node classification is achieved using GraphSAGE with the depth $K = 3$ (i.e., three hidden SAGEConv layers) and input feature dimension 128. The training configuration is shown in Table 17.

Table 18 Training configuration for GraphSAGE

Optimisation	Learning rate	alpha	eps	weight_decay	momentum	epochs
RMSProp	0.0001	0.99	1.00E-08	0.001	0.8	1000

The GraphSAGE models are trained based on semantic similarity from the context through inductive learning. The node classification performance on the test set through different pre-trained language models is shown in Figure 89. As can be seen, the text encoder from CLIP has the best performance for node layer identification, achieving 100% test accuracy for the BMKG with 10% - 35% missing links.

This experiment demonstrates that using the encoded text embeddings (through an LLM encoder) as node features can achieve excellent performance to identify the isolated nodes' layers in the incomplete BMKG after bidirectional DFS. The CLIP text encoder performs better than the other LLMs in this study for node classification, most likely attributed to its extensive training corpus. The result can enrich the BMKG semantic information at the node level and assist the subsequent link prediction.

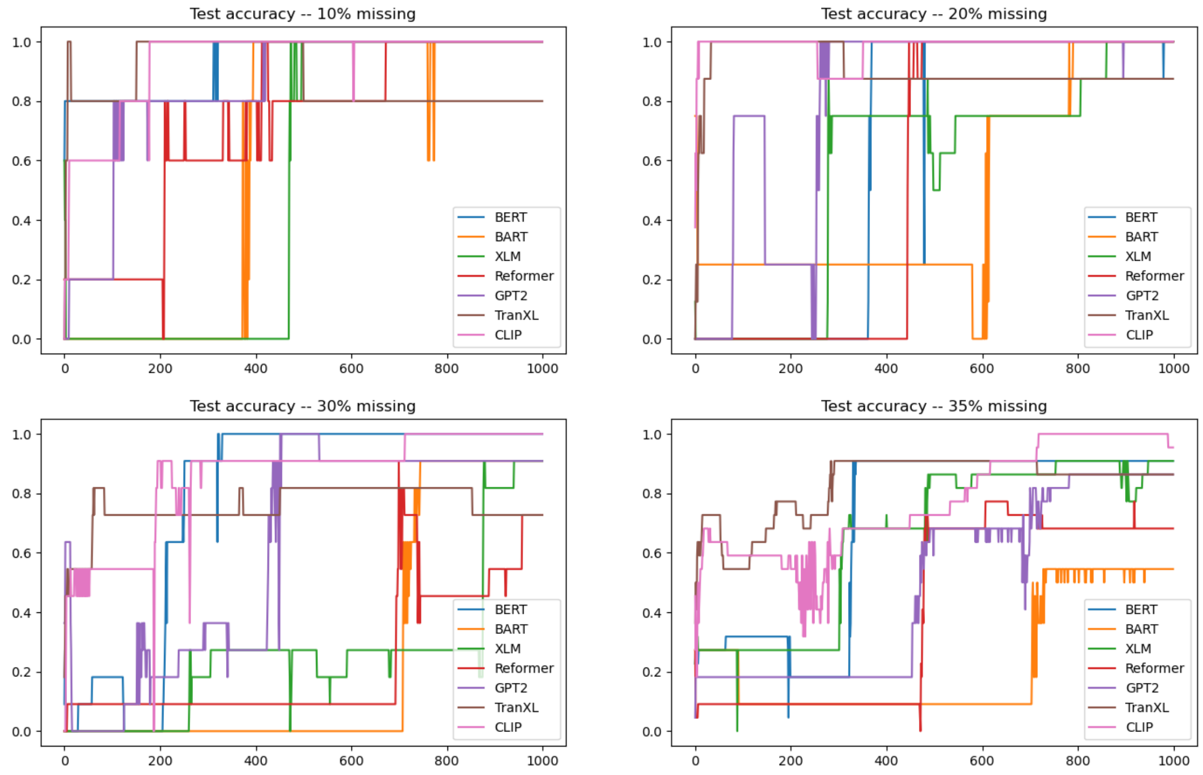


Figure 89 Node classification for incomplete BMKG with text encoding through LLMs

8.3.3 Maintenance recommendation

The experiment aims to predict the existence of edges between specific nodes in graphs following the pipeline in Section 4.2.3. The correct predicted links for BMKG completion can be accepted to suggest repair actions. The incomplete graphs are derived from the original BMKG (i.e., 48 nodes, 65 edges) by removing specific proportions of links (i.e., 10% ~ 40%). The utilized GraphSAGE in the experiment has three hidden SAGEConv layers, i.e., depth $K = 3$, based on mean aggregation and the input feature dimension is 128. The node features are generated through the pre-trained CLIP text encoder. The existing links (edges) and the missing links (edges) are taken as positive examples for the training and test sets, respectively. In contrast, the nonexistent links (edges) are adopted as negative examples and split into the training and test sets.

The pair-wise link scores are obtained by applying the dot product of node representations for each predicting edge. The same training configuration as in the previous node classification is adopted, shown in Table 17. The AUC on the test set is shown as the left columns in Figure 90, and the preliminary test accuracy is shown as the middle columns in Figure 90. Moreover, because the node links (i.e., pair connections) are strictly constrained between adjacent layers in the designed BMKG schema, the test accuracy can be further improved significantly by removing the unqualified links that violate the BMKG schema, shown as the right columns in Figure 90.

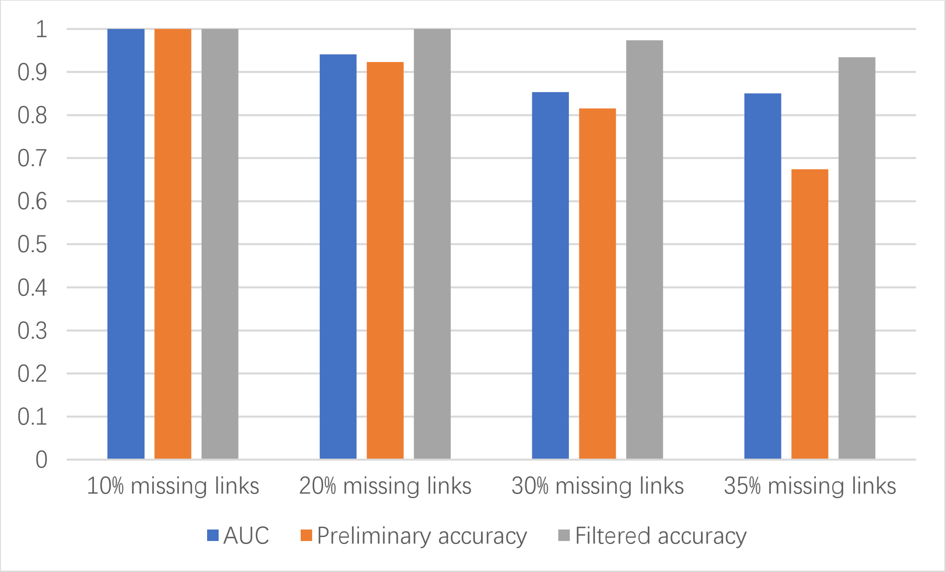


Figure 90 AUC and test accuracies of link prediction

In detail, by leveraging the aggregation and message propagation of GraphSAGE, when predicting if a link exists between the node pair, the trained model can consider the context of its upstream and downstream nodes, as shown in Figure 91. The real lines represent the existing links, and the dashed line indicates the predicted links. For example, directly linking (11) fixing to (25) failure in Figure 91 (a) is difficult. However, based on semantic similarity, the trained model can consider the subsequent node, i.e., (36) cut out fixing and ream the hole, to predict it. Similarly, the link between (14) stiffener angle and (27) distortion can also be predicted by considering the contextual similarity, i.e., (41) replace stiffener, as shown in Figure 91 (b).

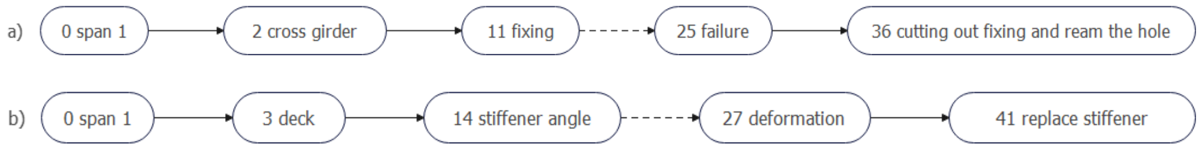
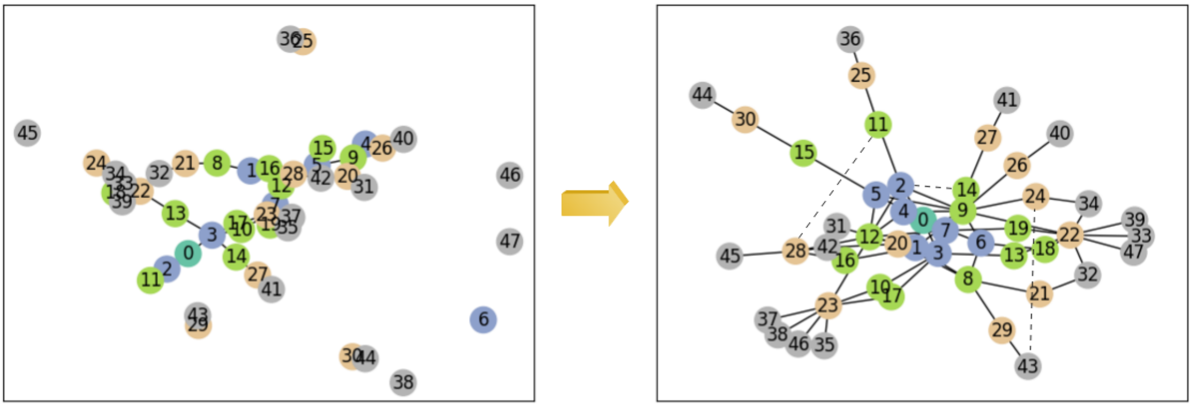


Figure 91 Contextual link prediction based on semantic similarity

The result of link prediction for an incomplete BMKG (35% missing links) is shown in Figure 92. Several extra links in prediction, which are non-existent in the current KMBG, are shown as dashed lines in Figure 92 (b). After careful examination by bridge maintenance experts, some of the extra predicted links are plausible, which had happened to other bridges or been observed in other maintenance reports, such as (2) cross girder → (14) stiffener angle, (24) hole → (43) fill with a similar product.



(a) Imcomplete BMKG (35% missing links)

(b) Intact BMKG with extra predicted links

Figure 92 Extra predicted links for incomplete BMKG with 35% missing links

This experiment demonstrates that the proposed approach via text encoding (through an LLM encoder) and contrastive learning can achieve excellent performance for link prediction. The model is learned from contextual similarity via inductive GraphSAGE through graph-based aggregation and message passing. The trained model can propose repair action for the unsolved defects from the existing options.

Finally, the proposed repair action can be forwarded to the qualified engineers for evaluation. If the engineer is satisfied with the suggestion, it will be added to maintenance planning and updated to the BMKG as a positive example. Otherwise, the pair link behind the suggestion will be updated to BMKG as a negative example to improve the training. The intact workflow is designed as shown in Figure 93.

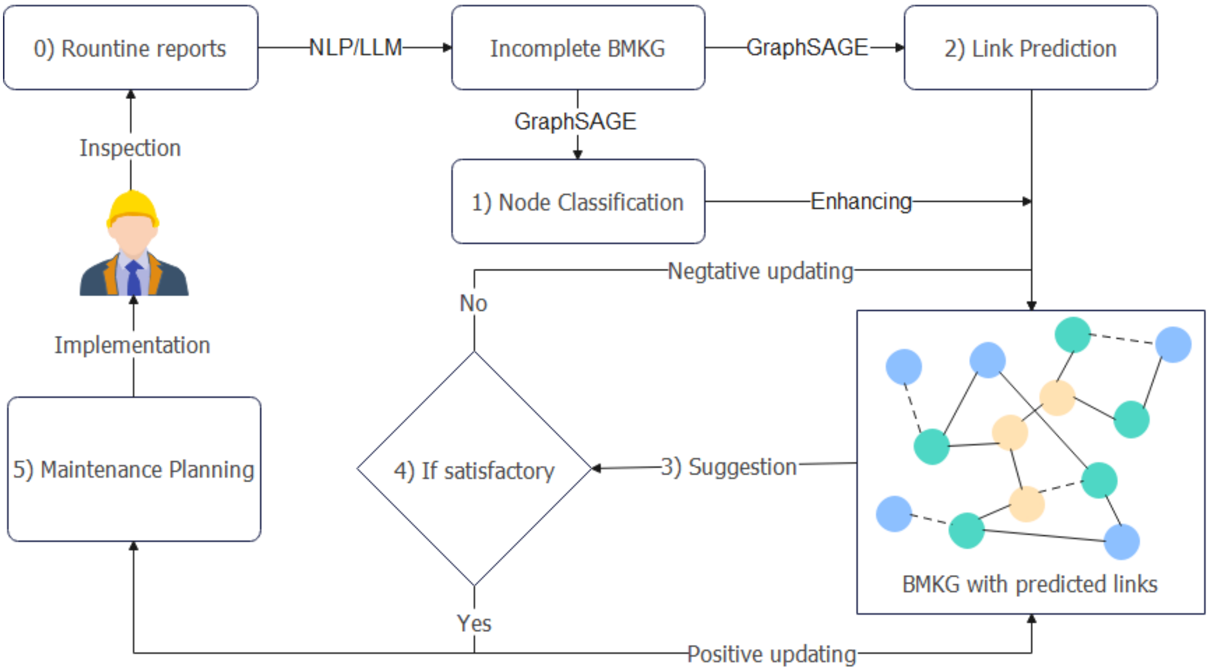


Figure 93 Designed workflow for AI-aided bridge maintenance planning by leveraging NLP, LLMs and GNNs

8.4 Summary

This work aims to explore graph data mining on BMKG, which shows superiority in documenting the domain expertise of bridge maintenance. Firstly, a BMKG schema is proposed based on bridge structure and practical maintenance reports, where the KG is oriented from top to bottom, and the node connection is strictly limited in adjacent layers. Then, the study explores contextual node classification and link prediction via GraphSAGE in the established BMKG, with text embeddings through LLMs as node features.

The experiment demonstrates that contextural node classification can identify the isolated nodes' layers in an incomplete BMKG (with missing links) after DFS, which is meaningful for node-level semantic enrichment. Then, contrastive learning via GraphSAGE can predict the missing links. The prediction accuracy can be enhanced significantly using the identified node layer information from the previous node classification. The link-prediction model is trained based on contextual similarity through inductive learning and can propose repair actions for unsolved defects from the existing options via an analogous strategy. Finally, an intact workflow (containing the proposed BMKG schema, feature embedding, and graph data mining approaches) is designed for bridge routine maintenance.

Although the proposed approach can achieve excellent performance for node layer identification and maintenance suggestion in an incomplete BMKG, it still has some limitations. Firstly, this preliminary work is based on a small knowledge graph with only 48 nodes and 66 edges. A large-scale knowledge graph based on a massive bridge maintenance corpus is required for further study. Secondly, the current model for maintenance recommendation can only provide repair proposals from the existing options. However, an existing option may not be the best solution for a specific defect. Therefore, more in-depth research to demonstrate the plausibility of triples and context based on the domain expertise is necessary. Moreover, generative models based on contextual understanding for bridge maintenance are required in the future.

Chapter 9 Bridge DT design and development

9.1 Overall design of system architecture

The bridge DT system architecture is designed as shown in Figure 94, consisting of three layers, i.e., presentation, application, and platform. The presentation layer includes data acquisition on the site of the physical bridge and the visualization of the virtual bridge at the used end. The heterogenous data input can be images, point clouds, time series, textural reports, etc. The user-end interface integrates the mainstream GIS and BIM web viewers, i.e., Cesium and Xeokit. The application layer involves usual bridge DT services (i.e., common APIs) for bridge inspection and assessment, such as damage detection and segmentation, FE and BIM model updating, structure health monitoring, as well as customized applications such as few-shot damage detection, optimization for maintenance planning, knowledge-based reasoning. Finally, the platform layer provides the back-end services for the web-based Digi-bridge platform, including essential web technologies, ML engine, big-data and cloud server.

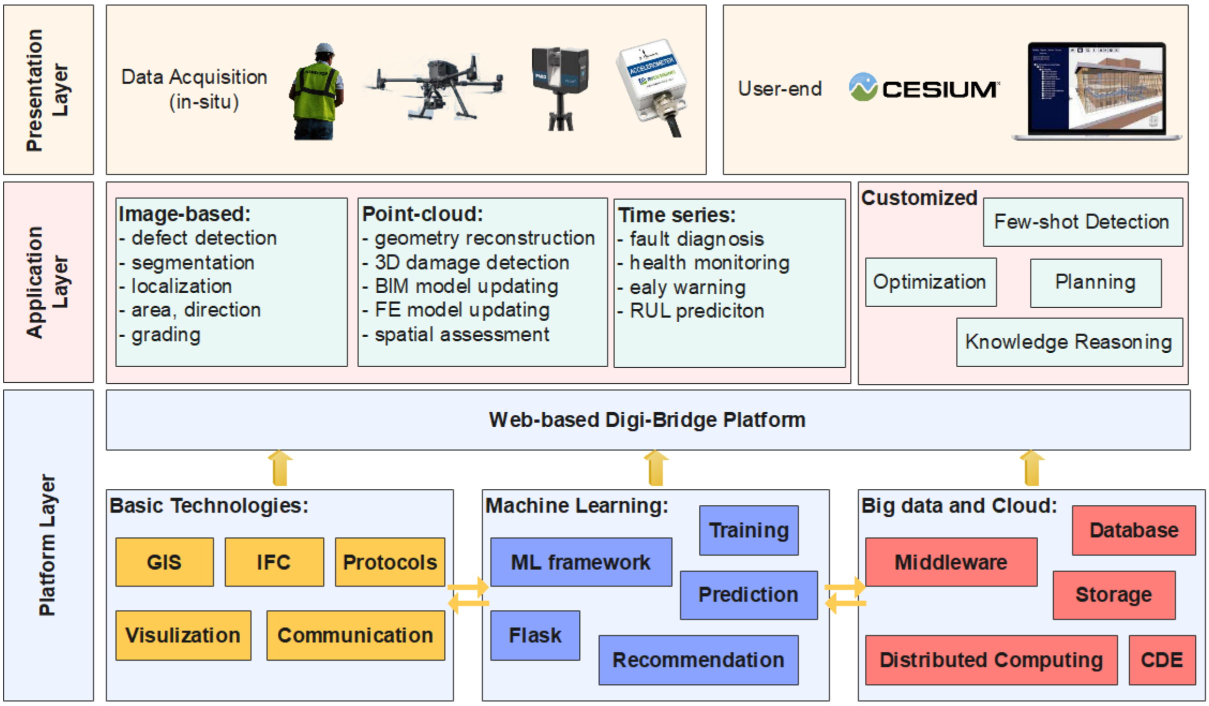


Figure 94 The overall architecture of the bridge DT system

9.2 Preliminary platform and DT services

9.2.1 Digi-bridge visualization and CDE

The complete Digi-bridge visualization pipeline and CDE interaction are shown in Figure 95. The CesiumJS presents multiple bridges and their surrounding geometric information with the coarse full bridge GLTF models. The weather data, prediction from the Met Office, and the local project information are integrated into the Cesium web page for enquiry. The detailed as-is built IFC model (LOD 500) is embedded on the web page via Xeokit SDK for BIM visualization. The monitoring sensors are pre-located on the corresponding entities (i.e., elements) and annotated according to the entity IDs from IfcBuildingElementProxy. Their data is visualized on the dashboard via Apache Echarts and updated from the IoT database. Similarly, the observed defects during the drone-enabled inspection are saved and linked to the corresponding elements based on the entity IDs for retrieval and evaluation. Therefore, an engineer can achieve remote monitoring and assessment through the developed digi-bridge platform in the office during the operation and inspection.

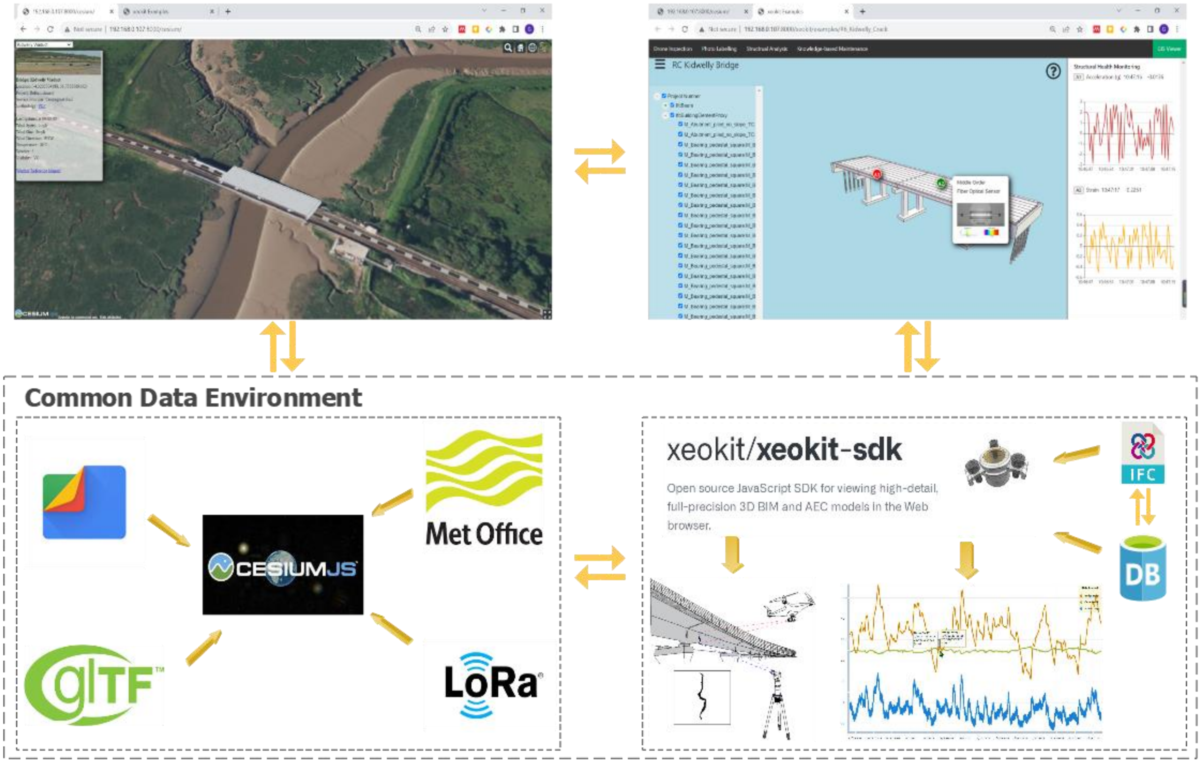
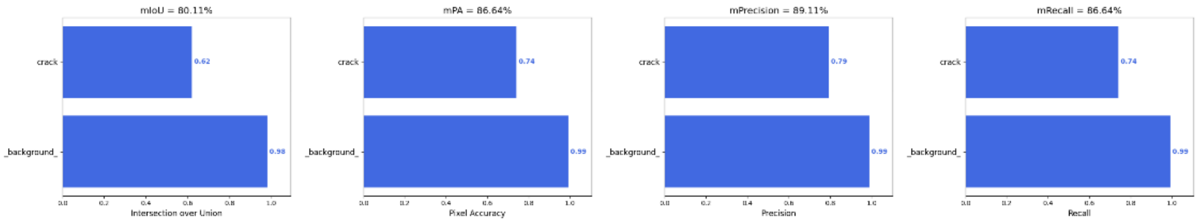

RC Kidwelly Bridge
Common Data Environment
CESIUMJS
Met Office
glTF
LoRa
xeokit/xeokit-sdk
Open source JavaScript SDK for viewing high-detail, full-precision 3D BIM and AEC models in the Web browser.
IFC
DB

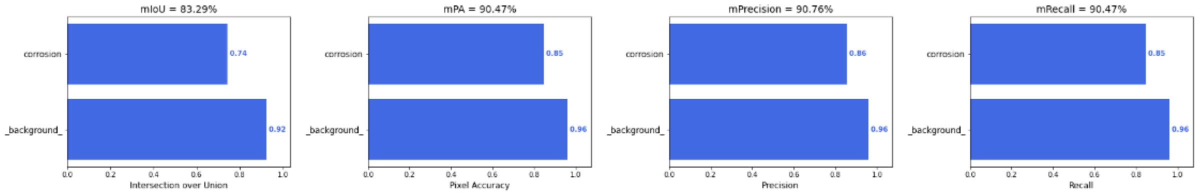
Figure 95 Digi-bridge visualization and CDE

9.2.2 Web-based API for damage segmentation

In the preliminary platform, the public COCO-bridge dataset [210] with annotated crack and corrosion images from practical bridge reports is utilized for the API development of crack and corrosion segmentation. The DeepLabV3+ architecture with the MobileNet-v2 backbone (pre-trained on the PASCAL VOC dataset [250]) is employed for model training, and the configuration is the same as in Table 8. The test results (i.e., mIoU, mPA, mPrecision, mRecall) are shown in Figure 96 (a) and (b).



(a) Crack segmentation on the test set



(a) Corrosion segmentation on the test set

Figure 96 Test performance of the trained DeepLabV3+ models

Then, the trained DeepLabV3+ models are deployed on the web server through the architecture shown in Figure 97. The API performance for crack and corrosion segmentation is illustrated in Figure 98. This application can support the quantitative damage evaluation during bridge inspection, such as cracking and corrosion. It also promises to be deployed on a drone based on its lightweight potential via knowledge distillation.

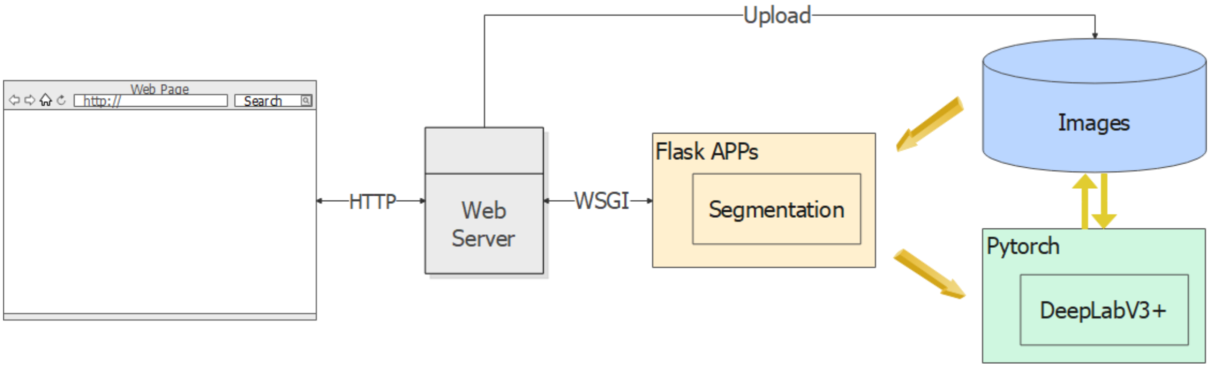
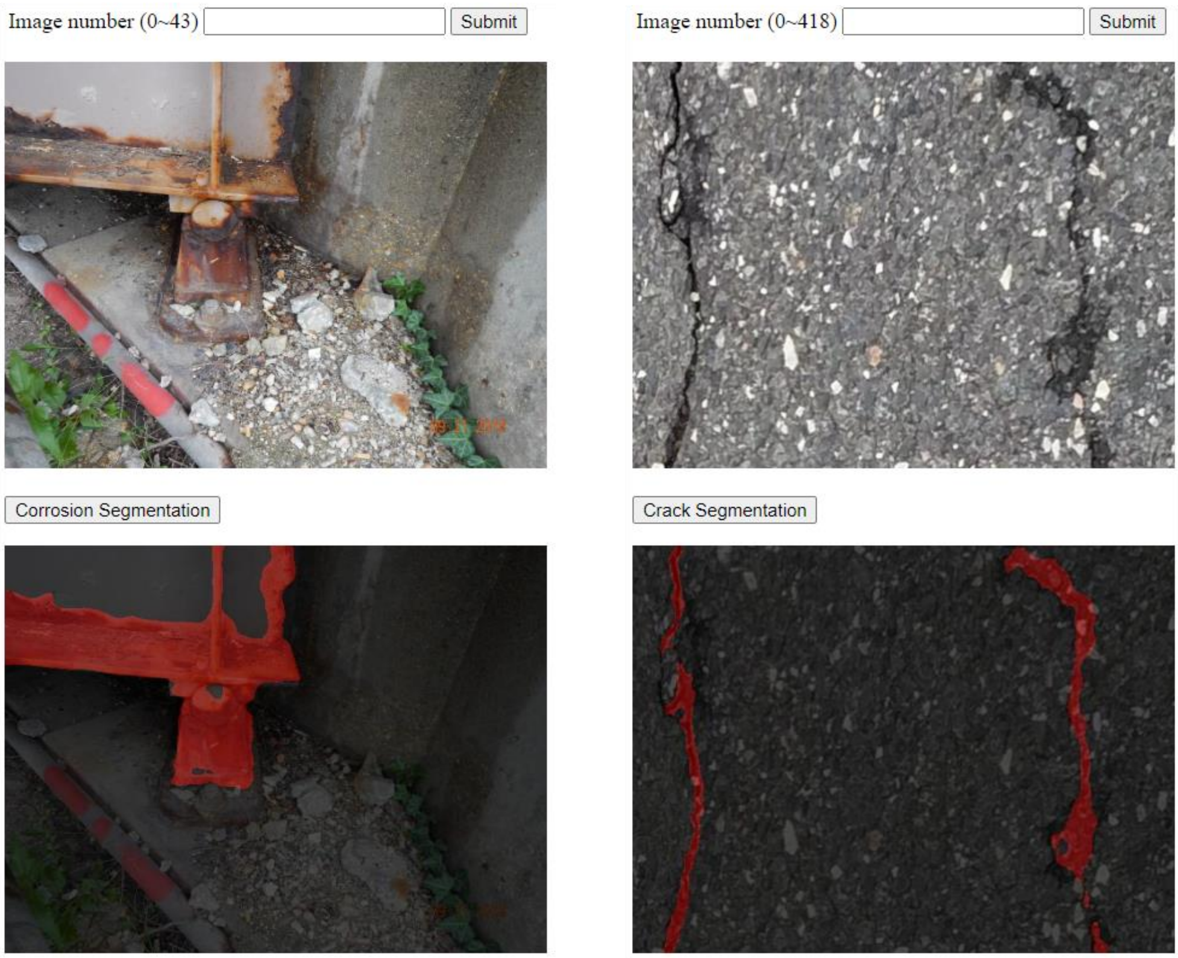


Figure 97 Web service architecture for damage segmentation



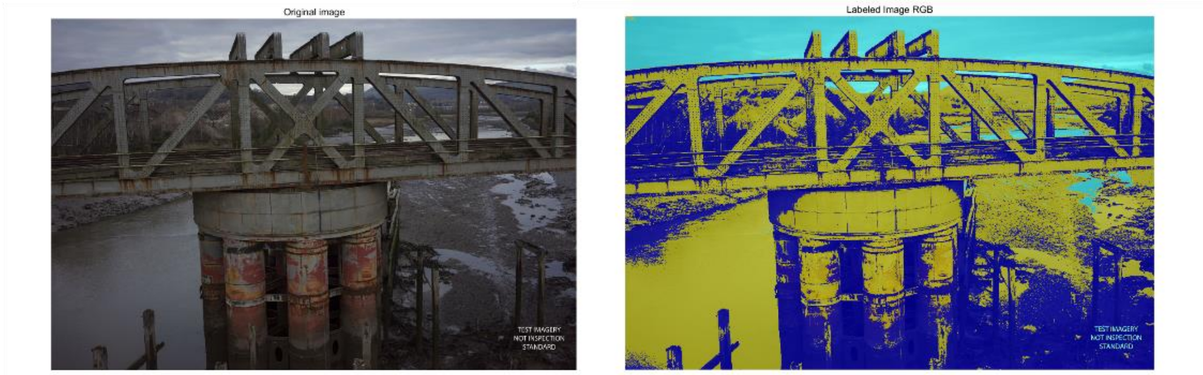
(a) Corrosion Segmentation

(a) Crack Segmentation

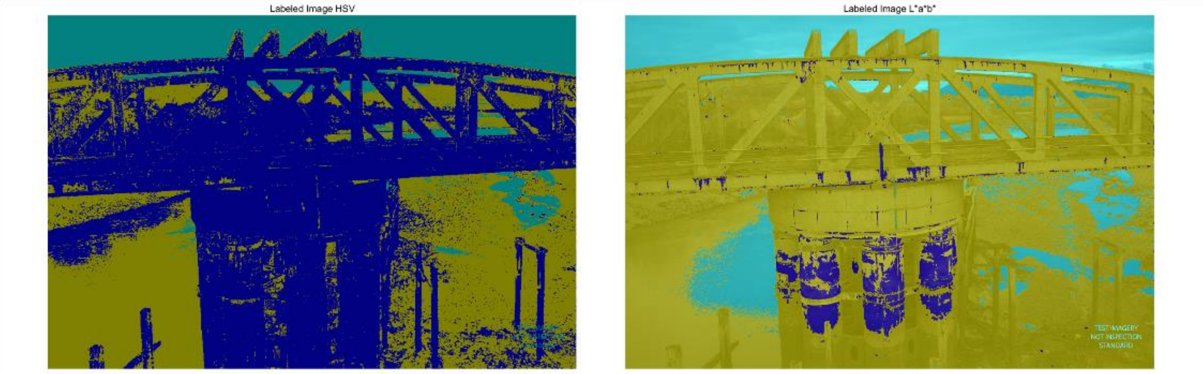
Figure 98 Web service for damage segmentation via DeepLabV3+

9.2.3 Unsupervised corrosion assessment

The unsupervised corrosion assessment is developed based on the practical inspection images from a pilot steel bridge. Three different colour spaces (i.e., RGB, HSV, L*a*b) and two unsupervised learning approaches (k-means and GMM) are utilised for comparison. The results demonstrate that L*a*b performs much better than the other two colour spaces for corrosion segmentation, i.e., the L*a*b clustering centres are more likely to indicate the foreground, background, and corrosion, as shown in Figure 99. Moreover, k-means and GMM have similar segmentation results.


Original image
Labeled Image RGB
TEST IMAGERY NOT INSPECTION STANDARD
TEST IMAGERY NOT INSPECTION STANDARD

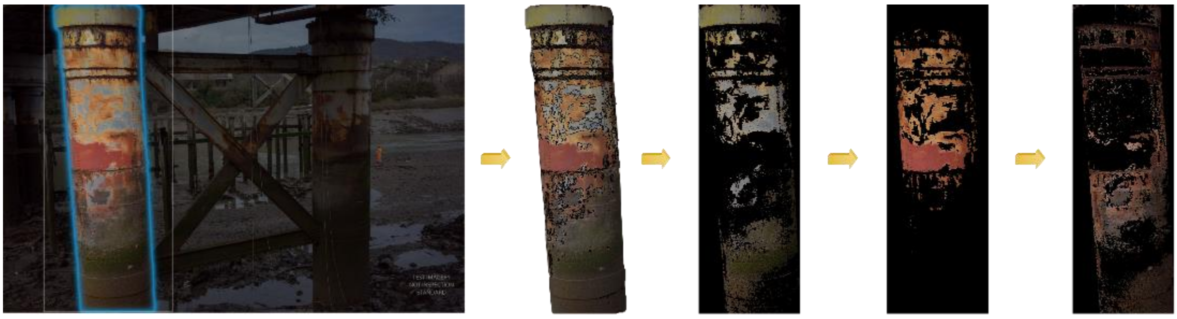
(a) Original Image (b) RGB (K=3)



(c) HSV (K=3) (d) L*a*b (K=3)

Figure 99 Corrosion segmentation via k-means in different colour spaces

Furthermore, the target structural element, such as a pier, can be segmented via the graph-cut algorithm or SAM for assessment. Then, the areas for each grade of corrosion are derived via k-means or GMM to support damage rating and repair decision-making by integrating super-pixels for downstream tasks, such as area labelling and calculation. The pipeline can be shown in Figure 100. The coating drop and fair rust area occupies 22% of the pier profile, which requires polish and repaint with anti-corrosion coating; the severe corrosion reaches 40%, which requires extra strengthening in addition to the above disposal.

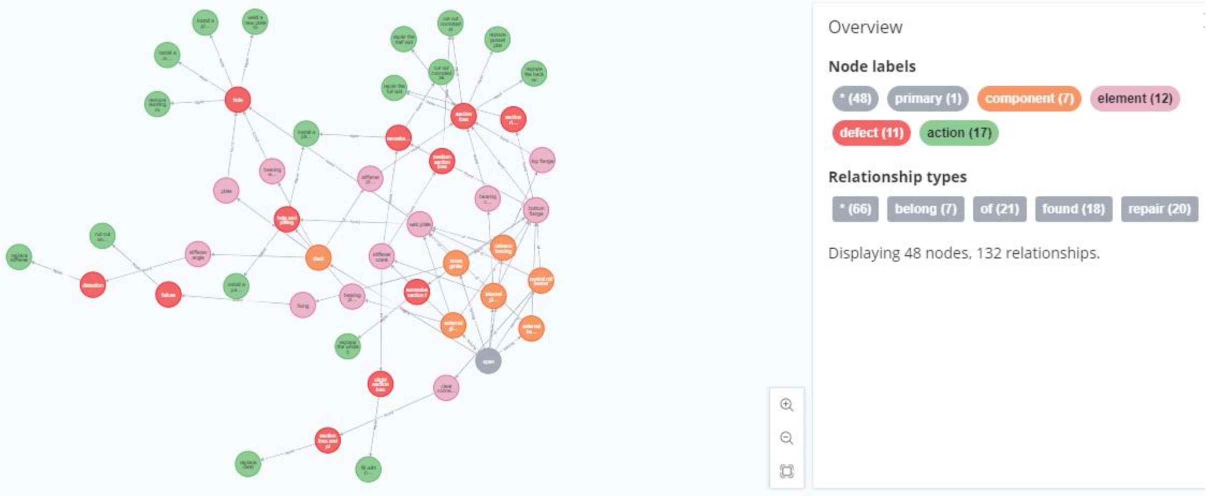


(a) Target element (b) Super-pixels (c) Normal (d) Coating drop and fair (e) Severe corrosion

Figure 100 Corrosion assessment via k-means in L*a*b

9.2.4 Question and answer chatbot

A graph dataset is created via Neo4j based on the BMKG with 48 nodes and 66 relationships in Section 8.3.1, as shown in Figure 101. Moreover, a preliminary chatbot for bridge inspection query is developed using spaCy (an open-source NLP library) and gradio interface.



Displaying 48 nodes, 132 relationships.

Figure 101 BMKG based on Neo4j

Four different sentence templates are designed for enquiry step by step as follows:

1) Question 1 – What components belong to A? A is at the primary structure layer, such as the span.

2) Question 2 – What elements does B have? B is at the component layer, such as the deck.

3) Question 3 – Is there any defect on C? C is at the element layer, such as the stiffener gusset plate.
4) Question 4 – How to repair D? D is at the defect layer, such as remarkable section loss.

The templated chatbot performance can be illustrated in Appendix B. Currently, it can only cover the maintenance query for the Neath River Swing Bridge based on a practical bridge inspection report from Centre Great Rail Ltd. The maintenance recommendations, such as analogous repair proposals like in Section 8.3.3, as well as versatile NLP functions and generative AI (such as the fine-tuning of ChatGPT), can be integrated into the chatbot in the future.

Chapter 10 Conclusion

Based on the research process and results, this Chapter aims to conclude the research by revisiting the research questions with rational answers point-by-point (Section 10.1), discussing the research limitations and future work (Section 10.2), as well as summarising the research contributions (Section 10.3).

10.1 Revisiting Research Questions

The pre-defined research questions and the corresponding answers are recapped below.

Question 1: How can an efficient, resilient, and bidirectional DT framework be created for practical bridge O&M in challenging conditions with massive heterogeneous data and limited communication?

The study in Chapter 4 proposes an AIoT-informed bridge DT framework to solve this question. The study indicates that the time delay of DT services consists of computation and communication time costs, depending on computational and communication complexity, respectively. It also reveals the distinct impact of their sequence on time consumption for DT services and the usage of information hierarchy to reduce communication complexity via AI-based edge computing. Then, bi-directional communication between edge and cloud is recommended to satisfy the restricted communication with minimal complexity for DT services. Moreover, AI-based edge computing can enable the system with resilience to endure a temporary loss of communication. Hence, a hierarchical communication architecture with fault-tolerant capability is designed based on LPWAN and mesh networks for different levels of DT services.

The proposed bridge DT framework is idealized mathematically, including state-space representation with time delay and inequalities for hardware processing capability. Meanwhile, the data flow and system resilience are demonstrated based on Petri-net modelling. Moreover, the framework is developed into a cross-platform prototype for bridge O&M and validated with different cases for bridge O&M, demonstrating the effectiveness, efficiency, and resilience of the proposed framework for bridge DTs.

Question 2: How can automatic bridge image-based defect detection be achieved under complex scenarios with weakly supervised information, such as very limited data?

The work in Chapter 5 proposes an approach based on improved ProtoNet for bridge damage detection under few-shot conditions for this question. The approach achieves feature embedding through cross-domain transfer learning from ImageNet, which enables embedding functions exempt from episodic training and become "training-free". Normalization is integrated into the improved ProtoNet to reduce the domain variation and enhance the transduction performance based on Euclidean distance. The linear classifier is added at the end of the backbone for classification, and fine-tuning based on the support set is further utilized to improve the performance.

An extensive ablation study is explored on a public bridge crack dataset, which proves that: (1) ImageNet is a reliable source domain for few-shot damage detection and can achieve a mean test accuracy of over 94% for 2-way 5-shot classification via the approach; (2) the proposed fine-tuning methods based on the FCN and Hadamard product perform better than the previous ones.

Finally, the approach is validated using real bridge inspection images, demonstrating its capability of fast implementation for damage detection with weakly supervised information and the potential for practical application in near real-time.

Question 3: How can the dense 3D point cloud data (such as from TLS) be integrated into the digital twinning pipeline efficiently, achieving DT model synchronization and volumetric assessment for local damages in near real-time?

The study in Chapter 6 proposes an efficient framework for 3D damage assessment and DT synchronization via point-cloud data. It starts from surface damage detection via DeepLabV3+ with only depth information and is further developed for spatial damage segmentation via morphological operations. Then, the separated point cloud is converted into a highly lightweight 3D binary matrix through voxelization and binarization, which can be compressed losslessly for downstream tasks.

The framework is validated via two case studies, demonstrating its superiority in volumetric assessment compared with the previous convex-hull fitting, as well as its excellent efficiency in geometric model updating of FEM and BIM through commercial software.

Question 4: How can intelligent fault diagnosis (IFD) and health-state monitoring be implemented through AutoML and data fusion (multi-axial or multi-sensor) based on time series?

The work in Chapter 7 proposes an efficient and unified framework leveraging AutoML and image-like data fusion for IFD with time-series signals from uniaxial or triaxial sensors. The data from a triaxial sensor can be reconstructed into a 3-channel pseudo-image by stacking the single-channel image from each axis to achieve data-level fusion. It can integrate the popular built-in and self-defined DL architectures, and the model training can be carried out consecutively or parallelly. Neural network search, hyperparameter optimization, and evaluation can be taken automatically based on comprehensive metrics. Finally, the recommended DL model can be easily deployed on a cloud server or an edge device via tinyML for inference to satisfy practical scenarios, such as in a DT system, which requires timely and resilient decision-making, even under communication-constraint circumstances.

The proposed framework and data fusion method are validated via two case studies based on uniaxial and triaxial vibration signals, respectively, demonstrating their efficiency for practical IFD application development and effectiveness for pattern recognition. Moreover, the fused triaxial time-series data through the proposed image-like data fusion method can improve the model performance effectively. Moreover, the recommended DL model can be easily deployed on a cloud server or an edge device (such as Raspberry Pi) via tinyML for inference to satisfy practical applications, such as in a DT manufacturing system, which requires timely and resilient decision-making, even under the communication-constraint circumstances.

As for health-state monitoring based on time-series signals from multiple sensors (more than three), a 1D-CNN architecture is proposed for data-level fusion and pattern recognition. The experiment demonstrates its effectiveness using a public dataset from the VBM project of a real-world railway bridge. Moreover, the data-level and decision-level fusion are compared, and the former performs much better in this dataset based on the proposed 1D-CNN architecture.

Question 5: How can we design a bridge maintenance knowledge graph (BMKG) based on bridge structures and practical inspection reports and then achieve maintenance recommendations via the analogical strategy?

The study in Chapter 8 proposes a BMKG schema based on bridge structure and practical maintenance reports. The KG architecture is oriented from top to bottom, and the node connection is strictly limited in adjacent layers. It shows superiority in documenting the domain knowledge of bridge maintenance. Then, the study explores graph data mining (including contextual node classification and link prediction) on the established BMKG by leveraging text encoding through LLMs and GNNs.

The experiment demonstrates that node classification can identify the isolated nodes' layers in an incomplete BMKG (with missing links) after DFS, which is meaningful for node-level semantic enrichment. Then, contrastive learning via inducive GraphSAGE can predict the missing links. The prediction accuracy can be enhanced using the identified node layer information from node classification. The trained link-prediction model can propose repair actions for unsolved defects from the existing options with an analogous strategy. Finally, an intact workflow (containing the proposed BMKG schema, feature embedding, and data mining approaches) is designed for bridge routine maintenance.

10.2 Research limitations and future work

The limitations and future work of this research are discussed below:

1. Although the proposed AIoT-informed bridge DT framework in Chapter 5 can significantly contribute to the efficiency and safety of bridge O&M, it still has some limitations. For example, AI-based edge computing raises a high requirement for power supply according to the tasks and algorithms, which can be an issue under resource-constraint circumstances. Hence, the edge AI in the study can only perform preliminary analysis and decision-making currently. Moreover, it reveals the great research significance of the sustainable power supply for edge devices and the trade-off between edge and cloud in data storage and computation.
2. Though the proposed FSL approach in Chapter 6 can contribute to automatic bridge defect detection under weakly supervised circumstances, it still has a few restrictions. Firstly, the approach is sensitive to noise, such as oil stains, road marks, shadows, and bridge joints. Hence, improving the approach's robustness is necessary in the next step. Secondly, the current approach only focuses on binary classification in fixed patches. However, multi-damage detection is usually required when different defects coexist in one image. The hierarchical ensemble learning and flexible region proposals are promising to help with this issue. Thirdly, the support examples should be representative across all the items for identification. However, selecting the support examples requires both ML and domain knowledge, which needs study in the future.
3. Chapter 7 proposes a highly lightweight and efficient framework which can contribute to 3D damage assessment and DT synchronization based on point clouds. However, it still has some limitations. The approach for 3D damage

detection in the framework is only suitable for damage perpendicular to a planar surface. The oblique (or twisted) damages are not available, as the LiDAR scanning cannot provide sufficient spatial information for those damages. Moreover, further developing this approach for spatial damages on a curved surface would be necessary in the next step. Furthermore, an appropriate point-cloud resolution is required when applying the proposed framework and approach, which relies on the survey equipment.

4. The proposed framework for health-state recognition by leveraging AutoML and data-level fusion in Chapter 8 can improve development for practical SHM applications. However, the proposed image-like fusion method is only suitable for a single triaxial sensor or three uniaxial sensors at the same sampling frequency. Therefore, fusion methods for heterogeneous data from multiple sensors with different sampling frequencies become interesting for future research. Moreover, the automatic model recommendation for edge devices by considering practical scenarios, such as computing capability, storage memory, and power supply, also has excellent research significance.
5. Although graph data mining by leveraging text encoding and GraphSAGE in Chapter 8 performs well for node layer identification and maintenance suggestion in an incomplete BMKG, it still has some limitations. Firstly, the established BMKG is based on a small knowledge graph derived from only several inspection reports, which cannot cover all the defect scenarios and repair actions. Future study requires a large-scale knowledge graph based on a massive bridge maintenance corpus. Secondly, the current maintenance recommendation can only provide repair proposals from the existing options, which may not be the best solution for an unseen defect. Therefore, future studies must delve into the triples and context based on the domain expertise. Additionally, generative models based on contextual understanding for bridge

maintenance would be helpful in future work.

10.3 Research contribution

This research contains works on several theoretical and practical developments to provide a data- and informatics-informed bridge DT system for a complete bridge maintenance cycle. By considering the findings and developments presented in the thesis, the main contributions of this research can be drawn as below:

1. Chapter 4 proposed and developed an AIoT-informed bridge DT framework in terms of mathematical idealization, Petri-net modelling, and LoRa-based prototype. Meanwhile, the study indicates that the time delay in a DT system consists of computation and communication time costs and reveals the distinct impact of their sequence on time consumption for DT services. Moreover, theoretically, information hierarchy and two-way communication (between edge and cloud) are leveraged and recommended to reduce communication complexity and satisfy the restricted communication with minimal complexity. Finally, AI-informed edge computing can enable the system with resilience to endure a temporary loss of communication, which is especially beneficial to the safety of physical bridges and public users.

   The above findings and developments were presented at the *29th EG-ICE International Workshop 2022* as a conference paper – "*An efficient and resilient digital-twin communication framework for smart bridge structural survey and maintenance"* and published as a journal article in *Automation in Construction*, entitled " *AIoT-informed digital twin communication for bridge maintenance*".

2. Chapter 5 proposed an FSL approach with an improved ProtoNet for automatic bridge visual inspection under weakly supervised circumstances. The designed few-shot damage detection framework and the proposed fine-

tuning method are validated under practical scenarios, demonstrating the approach's effectiveness and fast-implementation capability.

The above content was presented at the *30th EG-ICE International Workshop 2023* as a conference paper – “*Few-shot classification for image-based crack detection”* and published as a journal article in *Engineering Application of Artificial Intelligence*, entitled "*Few-shot learning for image-based bridge damage detection*”.

3. Chapter 6 proposed a highly efficient and lightweight framework for 3D damage evaluation and DT synchronization based on massive point-cloud data. The framework and related approaches are validated on synthetic and real-world damage scenarios, which can achieve volumetric assessment and model updating (for FE and BIM) in near real-time.

   The content of this study was written in a journal manuscript entitled “*Damage Volumetric Assessment and Digital Twin Synchronization Based on Point Clouds*” and submitted to *Automation in Construction* (minor revision).

4. Chapter 7 proposed an automatic and unified development framework for infrastructure health-state recognition based on DL by leveraging AutoML and data-level fusion. The framework is validated in different scenarios with the time-series data from a single sensor (including uniaxial and triaxial) or multiple sensors. At the same time, the comparative analysis reveals which method (between data-level and decision-level fusion) has better performance depending on different datasets and neural networks.

   The above findings and developments were published as a journal article in “*Machines*”, entitled “*A Deep Learning Framework for Intelligent Fault Diagnosis Using AutoML-CNN and Image-like Data Fusion*”.

5. Chapter 8 proposed a knowledge graph schema based on bridge structure

and practical inspection reports for bridge maintenance. Then, graph data mining through text encoding and GraphSAGE is successfully carried out on the established BMKG, i.e., contextual node classification for semantic enrichment and contrastive link prediction for maintenance recommendation. It demonstrates the availability of graph data mining on a knowledge graph for bridge maintenance by leveraging LLMs and GNNs. Finally, a detailed workflow is designed for bridge routine inspection practice, integrating the proposed KG schema and graph data mining approaches.

The content of this study was written as a conference paper – "*Bridge Maintenance Recommendation by Leveraging Large Language Models and Graph Neural Networks*", submitted to the *2024 ACM/SIGAPP Symposium on Applied Computing* (under review), as well as a journal manuscript entitled "*Exploring Bridge Maintenance Knowledge Graph by Leveraging Text Encoding and GraphSAGE*", submitted to *Automation in Construction* (under review).

References

[1] M. Grieves and J. Vickers, ‘Digital Twin: Mitigating Unpredictable, Undesirable Emergent Behavior in Complex Systems BT - Transdisciplinary Perspectives on Complex Systems: New Findings and Approaches’, F.-J. Kahlen, S. Flumerfelt, and A. Alves, Eds., Cham: Springer International Publishing, 2017, pp. 85–113. doi: 10.1007/978-3-319-38756-7_4.

[2] R. Sacks, I. Brilakis, E. Pikas, H. S. Xie, and M. Girolami, ‘Construction with digital twin information systems’, *Data-Centric Engineering*, vol. 1, no. 6, 2020, doi: 10.1017/dce.2020.16.

[3] Y. Lu, C. Liu, K. I. K. Wang, H. Huang, and X. Xu, ‘Digital Twin-driven smart manufacturing: Connotation, reference model, applications and research issues’, *Robot Comput Integr Manuf*, vol. 61, no. July 2019, p. 101837, 2020, doi: 10.1016/j.rcim.2019.101837.

[4] F. Tao, Q. Qi, L. Wang, and A. Y. C. Nee, ‘Digital Twins and Cyber–Physical Systems toward Smart Manufacturing and Industry 4.0: Correlation and Comparison’, *Engineering*, vol. 5, no. 4, pp. 653–661, 2019, doi: 10.1016/j.eng.2019.01.014.

[5] W. Kritzinger, M. Karner, G. Traar, J. Henjes, and W. Sihn, ‘Digital Twin in manufacturing: A categorical literature review and classification’, *IFAC-PapersOnLine*, vol. 51, no. 11, pp. 1016–1022, 2018, doi: 10.1016/j.ifacol.2018.08.474.

[6] A. J. H. Redelinghuys, A. H. Basson, and K. Kruger, ‘A six-layer architecture for the digital twin: a manufacturing case study implementation’, *J Intell Manuf*, vol. 31, no. 6, pp. 1383–1402, 2020, doi: 10.1007/s10845-019-01516-6.

[7] D. Jones, C. Snider, A. Nassehi, J. Yon, and B. Hicks, ‘Characterising the Digital Twin: A systematic literature review’, *CIRP J Manuf Sci Technol*, vol. 29, pp. 36–52, 2020, doi: 10.1016/j.cirpj.2020.02.002.

[8] C. Ye *et al.*, ‘A digital twin of bridges for structural health monitoring’, in *Structural Health Monitoring 2019: Enabling Intelligent Life-Cycle Health Management for Industry Internet of Things (IIOT) - Proceedings of the 12th International Workshop on Structural Health Monitoring*, 2019, pp. 1619–1626. doi: 10.12783/shm2019/32287.

[9] Y. Gao, H. Li, G. Xiong, and H. Song, ‘AIoT-informed digital twin communication for bridge maintenance’, *Autom Constr*, vol. 150, no. October 2022, p. 104835, 2023, doi: 10.1016/j.autcon.2023.104835.

[10] S. Yoon, S. Lee, S. Kye, I. H. Kim, H. J. Jung, and B. F. Spencer, ‘Seismic fragility analysis of deteriorated bridge structures employing a UAV inspection-based updated digital twin’, *Structural and Multidisciplinary Optimization*, vol. 65, no. 12, pp. 1–17, 2022, doi: 10.1007/s00158-022-03445-0.

[11] A. Adibfar and A. M. Costin, ‘Creation of a Mock-up Bridge Digital Twin by Fusing Intelligent Transportation Systems (ITS) Data into Bridge Information Model (BrIM)’, *J Constr Eng Manag*, vol. 148, no. 9, pp. 1–11, 2022, doi: 10.1061/(asce)co.1943-7862.0002332.

[12] M. Mohammadi, M. Rashidi, Y. Yu, and B. Samali, ‘Integration of TLS-derived Bridge Information Modeling (BrIM) with a Decision Support System (DSS) for digital twinning and asset management of bridge infrastructures’, *Comput Ind*, vol. 147, no. January, p. 103881, 2023, doi: 10.1016/j.compind.2023.103881.

[13] J. Yang *et al.*, ‘Intelligent bridge management via big data knowledge engineering’, *Autom Constr*, vol. 135, no. January 2021, p. 104118, 2022, doi: 10.1016/j.autcon.2021.104118.

[14] Z. Allah Bukhsh, I. Stipanovic, A. Saeed, and A. G. Doree, ‘Maintenance intervention predictions using entity-embedding neural networks’, *Autom Constr*, vol. 116, no. March, p. 103202, 2020, doi: 10.1016/j.autcon.2020.103202.

[15] B. Manzoor, I. Othman, and J. C. Pomares, ‘Digital technologies in the architecture, engineering and construction (Aec) industry—a bibliometric—qualitative literature review of research activities’, *Int J Environ Res Public Health*, vol. 18, no. 11, 2021, doi: 10.3390/ijerph18116135.

[16] D. G. J. Opoku, S. Perera, R. Osei-Kyei, and M. Rashidi, ‘Digital twin application in the construction industry: A literature review’, *Journal of Building Engineering*, vol. 40, no. February, p. 102726, 2021, doi: 10.1016/j.jobe.2021.102726.

[17] M. Li, X. Feng, and Y. Han, ‘Brillouin fiber optic sensors and mobile augmented reality-based digital twins for quantitative safety assessment of underground pipelines’, *Autom Constr*, vol. 144, no. March, p. 104617, 2022, doi: 10.1016/j.autcon.2022.104617.

[18] M. Li and X. Feng, ‘Multisensor data fusion-based structural health monitoring for buried metallic pipelines under complicated stress states’, *J Civ Struct Health Monit*, vol. 12, no. 6, pp. 1509–1521, 2022, doi: 10.1007/s13349-022-00609-w.

[19] N. S. Dang and C. S. Shim, ‘Bridge assessment for PSC girder bridge using digital twins model’, *Lecture Notes in Civil Engineering*, vol. 54, no. January, pp. 1241–1246, 2020, doi: 10.1007/978-981-15-0802-8_199.

[20] J. S. Kang, K. Chung, and E. J. Hong, ‘Multimedia knowledge-based bridge health monitoring using digital twin’, *Multimed Tools Appl*, vol. 80, no. 26–27, pp. 34609–34624, 2021, doi: 10.1007/s11042-021-10649-x.

[21] C. S. Shim, H. R. Kang, and N. S. Dang, ‘Digital twin models for maintenance of cable-supported bridges’, *International Conference on Smart Infrastructure and Construction 2019, ICSIC 2019: Driving Data-Informed Decision-Making*, no. January, pp. 737–742, 2019, doi: 10.1680/icsic.64669.737.

[22] J. Yang *et al.*, ‘Intelligent bridge management via big data knowledge engineering’, *Autom Constr*, vol. 135, p. 104118, 2022, doi: 10.1016/j.autcon.2021.104118.

[23] D. Kashmiri, F. Taherpour, M. Namian, and E. Ghiasvand, ‘Construction Research Congress 2020 809’, *Construction Research Congress*, vol. 2023, no. 1994, pp. 809–818, 2020.

[24] D. Dan, Y. Ying, and L. Ge, ‘Digital Twin System of Bridges Group Based on Machine Vision Fusion Monitoring of Bridge Traffic Load’, *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 11, pp. 22190–22205, 2021, doi: 10.1109/TITS.2021.3130025.

[25] D. Gürdür Broo, M. Bravo-Haro, and J. Schooling, ‘Design and implementation of a smart infrastructure digital twin’, *Autom Constr*, vol. 136, Apr. 2022, doi: 10.1016/j.autcon.2022.104171.

[26] H. Dang, M. Tatipamula, and H. X. Nguyen, ‘Cloud-Based Digital Twinning for Structural Health Monitoring Using Deep Learning’, *IEEE Trans Industr Inform*, vol. 18, no. 6, pp. 3820–3830, 2022, doi: 10.1109/TII.2021.3115119.

[27] C. Shim, N. Dang, S. Lon, and C. Jeon, ‘Development of a bridge maintenance system for prestressed concrete bridges using 3D digital twin model Development of a bridge maintenance system for prestressed concrete bridges using 3D digital twin model’, *Structure and Infrastructure Engineering*, vol. 15, no. 10, pp. 1319–1332, 2019, doi: 10.1080/15732479.2019.1620789.

[28] N. S. Dang, H. Kang, S. Lon, and C. S. Shim, ‘3D digital twin models for bridge maintenance’, *Proceedings of 10th International Conference on Short and Medium Span Bridges*, no. 73, pp. 1–9, 2018, [Online]. Available: https://www.researchgate.net/publication/331314334%0Ahttps://www.csce.ca/elf/apps/CONFERENCEVIEWER/conferences/SMSB/papers/FinalPaper_73_0508011616.doc

[29] ‘Hamburg Port Authority – smartBRIDGE Hamburg – World Port Sustainability Program’. Accessed: Feb. 21, 2023. [Online]. Available: https://sustainableworldports.org/project/hamburg-port-authority-smartbridge/

[30] K. Maes and G. Lombaert, ‘Monitoring Railway Bridge KW51 Before, During, and After Retrofitting’, *Journal of Bridge Engineering*, vol. 26, no. 3, p. 04721001, 2021, doi: 10.1061/(asce)be.1943-5592.0001668.

[31] S. O. Sajedi and X. Liang, ‘Vibration-based semantic damage segmentation for large-scale structural health monitoring’, *Computer-Aided Civil and Infrastructure Engineering*, vol. 35, no. 6, pp. 579–596, 2020, doi: 10.1111/mice.12523.

[32] C. Z. Dong, O. Celik, F. N. Catbas, E. J. O’Brien, and S. Taylor, ‘Structural displacement monitoring using deep learning-based full field optical flow methods’, *Structure and Infrastructure Engineering*, vol. 16, no. 1, pp. 51–71, 2020, doi:

10.1080/15732479.2019.1650078.

[33] X. Meng *et al.*, ‘Design and implementation of a new system for large bridge monitoring—geoshm’, *Sensors (Switzerland)*, vol. 18, no. 3, 2018, doi: 10.3390/s18030775.

[34] H. Alexakis, A. Franza, S. Acikgoz, and M. J. Dejong, ‘Monitoring bridge degradation using dynamic strain, acoustic emission and environmental data’, *International Conference on Smart Infrastructure and Construction 2019, ICSIC 2019: Driving Data-Informed Decision-Making*, vol. 2019, pp. 523–532, 2019, doi: 10.1680/icsic.64669.523.

[35] G. Ren, R. Ding, and H. Li, ‘Building an ontological knowledgebase for bridge maintenance’, *Advances in Engineering Software*, vol. 130, no. November 2018, pp. 24–40, 2019, doi: 10.1016/j.advengsoft.2019.02.001.

[36] J. Yang *et al.*, ‘Intelligent bridge management via big data knowledge engineering’, *Autom Constr*, vol. 135, no. November 2021, p. 104118, 2022, doi: 10.1016/j.autcon.2021.104118.

[37] R. Li, T. Mo, J. Yang, S. Jiang, T. Li, and Y. Liu, ‘Ontologies-Based Domain Knowledge Modeling and Heterogeneous Sensor Data Integration for Bridge Health Monitoring Systems’, *IEEE Trans Industr Inform*, vol. 17, no. 1, pp. 321–332, 2021, doi: 10.1109/TII.2020.2967561.

[38] H. Dang, M. Tatipamula, and H. X. Nguyen, ‘Cloud-Based Digital Twinning for Structural Health Monitoring Using Deep Learning’, *IEEE Trans Industr Inform*, vol. 18, no. 6, pp. 3820–3830, Jun. 2022, doi: 10.1109/TII.2021.3115119.

[39] J. Morris, D. Kroening, and P. Koopman, ‘Fault Tolerance Tradeoffs in Moving from Decentralized to Centralized Embedded Systems’, 2004.

[40] A. Helmrich, S. Markolf, R. Li, and T. Carvalhaes, ‘Centralization and decentralization for resilient infrastructure and complexity Centralization and decentralization for resilient infrastructure and complexity’, 2021.

[41] C. Ye, S. C. Kuok, L. J. Butler, and C. R. Middleton, ‘Implementing bridge model updating for operation and maintenance purposes: examination based on UK practitioners’ views’, *Structure and Infrastructure Engineering*, vol. 0, no. 0, pp. 1–20, 2021, doi: 10.1080/15732479.2021.1914115.

[42] P. Cawley, ‘Structural health monitoring: Closing the gap between research and industrial deployment’, *Struct Health Monit*, vol. 17, no. 5, pp. 1225–1244, Sep. 2018, doi: 10.1177/1475921717750047.

[43] H. Dang, M. Tatipamula, and H. X. Nguyen, ‘Cloud-Based Digital Twinning for Structural Health Monitoring Using Deep Learning’, *IEEE Trans Industr Inform*, vol. 18, no. 6, pp. 3820–3830, 2022, doi: 10.1109/TII.2021.3115119.

[44] A. Nair and C. S. Cai, 'Acoustic emission monitoring of bridges: Review and case studies', *Eng Struct*, vol. 32, no. 6, pp. 1704–1714, 2010, doi: 10.1016/j.engstruct.2010.02.020.

[45] N. Yousefpour, S. Downie, S. Walker, N. Perkins, and H. Dikanski, 'Machine Learning Solutions for Bridge Scour Forecast Based on Monitoring Data', *Transportation Research Record: Journal of the Transportation Research Board*, vol. 2675, no. 10, pp. 745–763, 2021, doi: 10.1177/03611981211012693.

[46] S. Lee, N. Kalos, and D. H. Shin, 'Non-destructive testing methods in the U.S. for bridge inspection and maintenance', *KSCE Journal of Civil Engineering*, vol. 18, no. 5, pp. 1322–1331, 2014, doi: 10.1007/s12205-014-0633-9.

[47] T. Omar and M. L. Nehdi, 'Thermal detection of subsurface delaminations in reinforced concrete bridge decks using unmanned aerial vehicle', *American Concrete Institute, ACI Special Publication*, vol. 2017-March, no. SP 331, pp. 1–14, 2017, doi: 10.14359/51715590.

[48] C. M. Yeum and S. J. Dyke, 'Vision-Based Automated Crack Detection for Bridge Inspection', *Computer-Aided Civil and Infrastructure Engineering*, vol. 30, no. 10, pp. 759–770, 2015, doi: 10.1111/mice.12141.

[49] C. Ye, S. Acikgoz, S. Pendrigh, E. Riley, and M. J. DeJong, 'Mapping deformations and inferring movements of masonry arch bridges using point cloud data', *Eng Struct*, vol. 173, no. July, pp. 530–545, 2018, doi: 10.1016/j.engstruct.2018.06.094.

[50] V. Barrile, G. Candela, A. Fotia, and E. Bernardo, 'UAV Survey of Bridges and Viaduct: Workflow and Application', *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11622 LNCS, pp. 269–284, 2019, doi: 10.1007/978-3-030-24305-0_21.

[51] S. T. Nguyen and H. M. La, 'A Climbing Robot for Steel Bridge Inspection', *Journal of Intelligent and Robotic Systems: Theory and Applications*, vol. 102, no. 4, pp. 1–21, 2021, doi: 10.1007/s10846-020-01266-1.

[52] B. Sutter *et al.*, 'A semi-autonomous mobile robot for bridge inspection', *Autom Constr*, vol. 91, no. May 2017, pp. 111–119, 2018, doi: 10.1016/j.autcon.2018.02.013.

[53] Y. Gao, H. Li, and G. Xiong, 'An Efficient and Resilient Digital-twin Communication Framework for Smart Bridge Structural Survey and Maintenance', in *Proceedings of the 29th EG-ICE International Workshop on Intelligent Computing in Engineering*, EG-ICE, Jun. 2022, pp. 165–175. doi: 10.7146/aul.455.c207.

[54] Y. Gao, H. Li, G. Xiong, and H. Song, 'AIoT-informed digital twin communication for bridge maintenance', *Autom Constr*, vol. 150, no. October 2022, p. 104835, 2023, doi: 10.1016/j.autcon.2023.104835.

[55] A. Flammini, P. Ferrari, D. Marioli, E. Sisinni, and A. Taroni, 'Wired and wireless sensor

networks for industrial applications', *Microelectronics J*, vol. 40, no. 9, pp. 1322–1336, 2009, doi: 10.1016/j.mejo.2008.08.012.

[56] L. Alonso, J. Barbarán, J. Chen, M. Díaz, L. Llopis, and B. Rubio, 'Middleware and communication technologies for structural health monitoring of critical infrastructures: A survey', *Comput Stand Interfaces*, vol. 56, no. March 2017, pp. 83–100, 2018, doi: 10.1016/j.csi.2017.09.007.

[57] B. Foubert and N. Mitton, 'Long-range wireless radio technologies: A survey', *Future Internet*, vol. 12, no. 1, 2020, doi: 10.3390/fi12010013.

[58] K. Mekki, E. Bajic, F. Chaxel, and F. Meyer, 'A comparative study of LPWAN technologies for large-scale IoT deployment', *ICT Express*, vol. 5, no. 1, pp. 1–7, 2019, doi: 10.1016/j.icte.2017.12.005.

[59] S. M. Oh, Y. J. Kim, C. S. Park, and I. H. Kim, 'Process-driven BIM-based optimal design using integration of EnergyPlus, genetic algorithm, and Pareto optimality', *Proceedings of Building Simulation 2011: 12th Conference of International Building Performance Simulation Association*, pp. 894–901, 2011.

[60] K. Afsari, L. Florez, E. Maneke, and M. Afkhamiaghda, 'An experimental investigation of the integration of smart building components with building information model (BIM)', *Proceedings of the 36th International Symposium on Automation and Robotics in Construction, ISARC 2019*, no. Isarc, pp. 578–585, 2019, doi: 10.22260/isarc2019/0077.

[61] J. M. D. Delgado, L. J. Butler, N. Gibbons, I. Brilakis, M. Z. E. B. Elshafie, and C. Middleton, 'Management of structural monitoring data of bridges using BIM', *Proceedings of the Institution of Civil Engineers: Bridge Engineering*, vol. 170, no. 3, pp. 204–218, 2017, doi: 10.1680/jbren.16.00013.

[62] Y. Xu and Y. Turkan, 'BrIM and UAS for bridge inspections and management', *Engineering, Construction and Architectural Management*, vol. 27, no. 3, pp. 785–807, 2020, doi: 10.1108/ECAM-12-2018-0556.

[63] P. Singh and A. Sadhu, 'System Identification-Enhanced Visualization Tool for Infrastructure Monitoring and Maintenance', *Front Built Environ*, vol. 6, no. May, 2020, doi: 10.3389/fbuil.2020.00076.

[64] C. Boddupalli, A. Sadhu, E. Rezazadeh Azar, and S. Pattyson, 'Improved visualization of infrastructure monitoring data using building information modeling', *Structure and Infrastructure Engineering*, vol. 15, no. 9, pp. 1247–1263, 2019, doi: 10.1080/15732479.2019.1602150.

[65] J. M. Davila Delgado, L. J. Butler, I. Brilakis, M. Z. E. B. Elshafie, and C. R. Middleton, 'Structural Performance Monitoring Using a Dynamic Data-Driven BIM Environment', *Journal of Computing in Civil Engineering*, vol. 32, no. 3, p. 04018009, 2018, doi: 10.1061/(asce)cp.1943-5487.0000749.

[66] F. Diara and F. Rinaudo, ‘Ark-bim: Open-source cloud-based hbim platform for archaeology’, *Applied Sciences (Switzerland)*, vol. 11, no. 18, Sep. 2021, doi: 10.3390/app11188770.

[67] L. Chamari, E. Petrova, and P. Pauwels, ‘A web-based approach to BMS, BIM and IoT integration: a case study’, doi: 10.34641/clima.2022.228.

[68] Y. Zhang and J. Beetz, ‘Building-CPS: Cyber-Physical System for Building Environment Monitoring DuraArk View project Building-CPS: Cyber-Physical System for Building Environment Monitoring’. [Online]. Available: https://www.researchgate.net/publication/355456670

[69] Y. M. Hsieh and Y. S. Lu, ‘Visualization system for field monitoring data and its effectiveness’, *Autom Constr*, vol. 26, pp. 54–68, 2012, doi: 10.1016/j.autcon.2012.03.004.

[70] A. Khudhair, H. Li, T. Bower, and G. Ren, ‘A theoretical holistic decision-making framework supporting collaborative design based on common data analysis (CDA) method’, *Journal of Building Engineering*, vol. 46, no. November, p. 103686, 2022, doi: 10.1016/j.jobe.2021.103686.

[71] L. Sun, Z. Shang, Y. Xia, S. Bhowmick, and S. Nagarajaiah, ‘Review of Bridge Structural Health Monitoring Aided by Big Data and Artificial Intelligence: From Condition Assessment to Damage Detection’, *Journal of Structural Engineering*, vol. 146, no. 5, p. 04020073, May 2020, doi: 10.1061/(asce)st.1943-541x.0002535.

[72] C. H. J. Duy Cuong Nguyen, Chang Su Shim, The Quan Nguyen, Ruoyu Jin, ‘Developing a mixed-reality based application for bridge inspection and maintenance’, *Enabling The Development And Implementation of Digital Twins - Proceedings of the 20th International Conference on Construction Applications of Virtual Reality - (30th Sep - 2nd Oct 2020)*, pp. 31–43, 2020.

[73] M. Omer, L. Margetts, M. Hadi Mosleh, S. Hewitt, and M. Parwaiz, ‘Use of gaming technology to bring bridge inspection to the office’, *Structure and Infrastructure Engineering*, vol. 15, no. 10, pp. 1292–1307, 2019, doi: 10.1080/15732479.2019.1615962.

[74] Y. Li, M. M. Karim, and R. Qin, ‘A Virtual-Reality-Based Training and Assessment System for Bridge Inspectors With an Assistant Drone’, *IEEE Trans Hum Mach Syst*, pp. 1–11, 2022, doi: 10.1109/thms.2022.3155373.

[75] S. N. Sakib, T. Ane, N. Matin, and M. S. Kaiser, ‘An intelligent flood monitoring system for Bangladesh using wireless sensor network’, *2016 5th International Conference on Informatics, Electronics and Vision, ICIEV 2016*, pp. 979–984, 2016, doi: 10.1109/ICIEV.2016.7760145.

[76] E. Leon, C. Alberoni, M. Wister, and J. Hernández-Nolasco, ‘Flood Early Warning

System by Twitter Using LoRa', p. 1213, 2018, doi: 10.3390/proceedings2191213.

[77] I. Srikanth and M. Arockiasamy, 'Deterioration models for prediction of remaining useful life of timber and concrete bridges: A review', *Journal of Traffic and Transportation Engineering (English Edition)*, vol. 7, no. 2, pp. 152–173, 2020, doi: 10.1016/j.jtte.2019.09.005.

[78] Y. Goi and C. W. Kim, 'Damage detection of a truss bridge utilizing a damage indicator from multivariate autoregressive model', *J Civ Struct Health Monit*, vol. 7, no. 2, pp. 153–162, 2017, doi: 10.1007/s13349-017-0222-y.

[79] X. X. Cheng, J. H. Fan, and Z. H. Xiao, 'Finite element model updating for the Tsing Ma Bridge tower based on surrogate models', *Journal of Low Frequency Noise Vibration and Active Control*, vol. 41, no. 2, pp. 500–518, 2022, doi: 10.1177/14613484211058999.

[80] S. O. Sajedi and X. Liang, 'Vibration-based semantic damage segmentation for large-scale structural health monitoring', *Computer-Aided Civil and Infrastructure Engineering*, vol. 35, no. 6, pp. 579–596, 2020, doi: 10.1111/mice.12523.

[81] Z. Allah Bukhsh, I. Stipanovic, A. Saeed, and A. G. Doree, 'Maintenance intervention predictions using entity-embedding neural networks', *Autom Constr*, vol. 116, no. March, p. 103202, 2020, doi: 10.1016/j.autcon.2020.103202.

[82] R. Li, T. Mo, J. Yang, S. Jiang, T. Li, and Y. Liu, 'Ontologies-Based Domain Knowledge Modeling and Heterogeneous Sensor Data Integration for Bridge Health Monitoring Systems', *IEEE Trans Industr Inform*, vol. 17, no. 1, pp. 321–332, 2021, doi: 10.1109/TII.2020.2967561.

[83] J. König, M. Jenkins, M. Mannion, P. Barrie, and G. Morison, 'What's Cracking? A Review and Analysis of Deep Learning Methods for Structural Crack Segmentation, Detection and Quantification', pp. 1–18, 2022.

[84] T.-Y. Lin *et al.*, 'Microsoft coco: Common objects in context', in *European conference on computer vision*, Springer, 2014, pp. 740–755.

[85] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, 'The pascal visual object classes (voc) challenge', *Int J Comput Vis*, vol. 88, no. 2, pp. 303–338, 2010.

[86] Y. J. Cha, W. Choi, and O. Büyüköztürk, 'Deep Learning-Based Crack Damage Detection Using Convolutional Neural Networks', *Computer-Aided Civil and Infrastructure Engineering*, vol. 32, no. 5, pp. 361–378, 2017, doi: 10.1111/mice.12263.

[87] H. Fu, D. Meng, W. Li, and Y. Wang, 'Bridge crack semantic segmentation based on improved deeplabv3+', *J Mar Sci Eng*, vol. 9, no. 6, 2021, doi: 10.3390/jmse9060671.

[88] S. Wang, S. Qiu, W. Wang, D. Xiao, and K. C. P. Wang, 'Cracking Classification Using Minimum Rectangular Cover–Based Support Vector Machine', *Journal of Computing*

in Civil Engineering, vol. 31, no. 5, pp. 1–9, 2017, doi: 10.1061/(asce)cp.1943-5487.0000672.

[89] Y. Fujita, K. Shimada, M. Ichihara, and Y. Hamamoto, ‘A method based on machine learning using hand-crafted features for crack detection from asphalt pavement surface images’, *Thirteenth International Conference on Quality Control by Artificial Vision 2017*, vol. 10338, p. 103380I, 2017, doi: 10.1117/12.2264075.

[90] F. C. Chen, M. R. Jahanshahi, R. T. Wu, and C. Joffe, ‘A texture-Based Video Processing Methodology Using Bayesian Data Fusion for Autonomous Crack Detection on Metallic Surfaces’, *Computer-Aided Civil and Infrastructure Engineering*, vol. 32, no. 4, pp. 271–287, 2017, doi: 10.1111/mice.12256.

[91] S. Wang, X. Liu, T. Yang, and X. Wu, ‘Panoramic Crack Detection for Steel Beam Based on Structured Random Forests’, *IEEE Access*, vol. 6, pp. 16432–16444, 2018, doi: 10.1109/ACCESS.2018.2812141.

[92] Y. Shi, L. Cui, Z. Qi, F. Meng, and Z. Chen, ‘Automatic Road Crack Detection Using Random Structured Forests’, *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 12, pp. 1–12, 2016, doi: 10.1109/TITS.2016.2552248.

[93] Q. Luo, B. Ge, and Q. Tian, ‘A fast adaptive crack detection algorithm based on a double-edge extraction operator of FSM’, *Constr Build Mater*, vol. 204, pp. 244–254, 2019, doi: 10.1016/j.conbuildmat.2019.01.150.

[94] D. Frias and J. Hidalgo, ‘A High Accuracy Image Hashing and Random Forest Classifier for Crack Detection in Concrete Surface Images’, pp. 1–13, 2021, [Online]. Available: http://arxiv.org/abs/2106.05755

[95] S. Wang, F. Yang, Y. Cheng, Y. Yang, and Y. Wang, ‘Adaboost-based Crack Detection Method for Pavement’, *IOP Conf Ser Earth Environ Sci*, vol. 189, no. 2, 2018, doi: 10.1088/1755-1315/189/2/022005.

[96] A. Cord and S. Chambon, ‘Automatic Road Defect Detection by Textural Pattern Recognition Based on AdaBoost’, *Computer-Aided Civil and Infrastructure Engineering*, vol. 27, no. 4, pp. 244–259, 2012, doi: 10.1111/j.1467-8667.2011.00736.x.

[97] L. Wang, L. Zhuang, and Z. Zhang, ‘Automatic Detection of Rail Surface Cracks with a Superpixel-Based Data-Driven Framework’, *Journal of Computing in Civil Engineering*, vol. 33, no. 1, pp. 1–9, 2019, doi: 10.1061/(asce)cp.1943-5487.0000799.

[98] H. G. Moon and J. H. Kim, ‘Inteligent crack detecting algorithm on the concrete crack image using neural network’, *Proceedings of the 28th International Symposium on Automation and Robotics in Construction, ISARC 2011*, pp. 1461–1467, 2011, doi: 10.22260/isarc2011/0279.

[99] N. D. Hoang, ‘An Artificial Intelligence Method for Asphalt Pavement Pothole Detection Using Least Squares Support Vector Machine and Neural Network with

Steerable Filter-Based Feature Extraction', *Advances in Civil Engineering*, vol. 2018, 2018, doi: 10.1155/2018/7419058.

[100] H. D. Cheng, J. Wang, Y. G. Hu, C. Glazier, X. J. Shi, and X. W. Chen, 'Novel approach to pavement cracking detection based on neural network', *Transp Res Rec*, no. 1764, pp. 119–127, 2001, doi: 10.3141/1764-13.

[101] Y.-A. Hsieh and Y. J. Tsai, 'Machine Learning for Crack Detection: Review and Model Performance Comparison', *Journal of Computing in Civil Engineering*, vol. 34, no. 5, pp. 1–12, 2020, doi: 10.1061/(asce)cp.1943-5487.0000918.

[102] F. C. Chen, M. R. Jahanshahi, R. T. Wu, and C. Joffe, 'A texture-Based Video Processing Methodology Using Bayesian Data Fusion for Autonomous Crack Detection on Metallic Surfaces', *Computer-Aided Civil and Infrastructure Engineering*, vol. 32, no. 4, pp. 271–287, 2017, doi: 10.1111/mice.12256.

[103] S. Wang, S. Qiu, W. Wang, D. Xiao, and K. C. P. Wang, 'Cracking Classification Using Minimum Rectangular Cover–Based Support Vector Machine', *Journal of Computing in Civil Engineering*, vol. 31, no. 5, pp. 1–9, 2017, doi: 10.1061/(asce)cp.1943-5487.0000672.

[104] P. Prasanna *et al.*, 'Automated Crack Detection on Concrete Bridges', *IEEE Transactions on Automation Science and Engineering*, vol. 13, no. 2, pp. 591–599, 2016, doi: 10.1109/TASE.2014.2354314.

[105] Y. Mohammed, N. Uddin, C. Tan, and Z. Shi, 'Crack Detection using Faster R-CNN and Point Feature Matching', vol. 10, no. 3, 2020, doi: 10.19080/CERJ.2020.10.555790.

[106] M. Nie and C. Wang, 'Pavement Crack Detection based on yolo v3', *Proceedings - 2019 2nd International Conference on Safety Produce Informatization, IICSPI 2019*, pp. 327–330, 2019, doi: 10.1109/IICSPI48186.2019.9095956.

[107] H. Xu, X. Su, Y. Wang, H. Cai, K. Cui, and X. Chen, 'Automatic bridge crack detection using a convolutional neural network', *Applied Sciences (Switzerland)*, vol. 9, no. 14, 2019, doi: 10.3390/app9142867.

[108] S. Wu, J. Fang, X. Zheng, and X. Li, 'Sample and Structure-Guided Network for Road Crack Detection', *IEEE Access*, vol. 7, pp. 130032–130043, 2019, doi: 10.1109/ACCESS.2019.2940767.

[109] H. Liu, X. Miao, C. Mertz, C. Xu, and H. Kong, 'CrackFormer: Transformer Network for Fine-Grained Crack Detection', *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3763–3772, 2021, doi: 10.1109/ICCV48922.2021.00376.

[110] J. Fang, C. Yang, Y. Shi, N. Wang, and Y. Zhao, 'External Attention Based TransUNet and Label Expansion Strategy for Crack Detection', *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 10, pp. 19054–19063, 2022, doi: 10.1109/TITS.2022.3154407.

[111] Y. Pan, G. Zhang, and L. Zhang, ‘A spatial-channel hierarchical deep learning network for pixel-level automated crack detection’, *Autom Constr*, vol. 119, no. July, p. 103357, 2020, doi: 10.1016/j.autcon.2020.103357.

[112] X. Xiang, Z. Wang, and Y. Qiao, ‘An Improved YOLOv5 Crack Detection Method Combined with Transformer’, *IEEE Sens J*, vol. 22, no. 14, pp. 14328–14335, 2022, doi: 10.1109/JSEN.2022.3181003.

[113] Y. J. Cha, W. Choi, G. Suh, S. Mahmoudkhani, and O. Büyüköztürk, ‘Autonomous Structural Visual Inspection Using Region-Based Deep Learning for Detecting Multiple Damage Types’, *Computer-Aided Civil and Infrastructure Engineering*, vol. 33, no. 9, pp. 731–747, 2018, doi: 10.1111/mice.12334.

[114] M. Mundt, S. Majumder, S. Murali, P. Panetsos, and V. Ramesh, ‘Meta-learning convolutional neural architectures for multi-target concrete defect classification with the concrete defect bridge image dataset’, *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2019-June, pp. 11188–11197, 2019, doi: 10.1109/CVPR.2019.01145.

[115] Y. Xu, Y. Bao, Y. Zhang, and H. Li, ‘Attribute-based structural damage identification by few-shot meta learning with inter-class knowledge transfer’, *Struct Health Monit*, vol. 20, no. 4, pp. 1494–1517, 2021, doi: 10.1177/1475921720921135.

[116] V. Vapnik, *The nature of statistical learning theory*. Springer science & business media, 1999.

[117] B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum, ‘The Omniglot challenge: a 3-year progress report’, *Curr Opin Behav Sci*, vol. 29, pp. 97–104, 2019, doi: 10.1016/j.cobeha.2019.04.007.

[118] L. Bertinetto, P. H. S. Torr, J. Henriques, and A. Vedaldi, ‘Meta-learning with differentiable closed-form solvers’, *7th International Conference on Learning Representations, ICLR 2019*, pp. 1–15, 2019.

[119] E. Triantafillou *et al.*, ‘Meta-dataset: A dataset of datasets for learning to learn from few examples’, *arXiv preprint arXiv:1903.03096*, 2019.

[120] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie, ‘The caltech-ucsd birds-200-2011 dataset’, 2011.

[121] G. Hu, H. Wu, Y. Zhang, and M. Wan, ‘A low shot learning method for tea leaf’s disease identification’, *Comput Electron Agric*, vol. 163, no. May, 2019, doi: 10.1016/j.compag.2019.104852.

[122] B. Hariharan and R. Girshick, ‘Low-Shot Visual Recognition by Shrinking and Hallucinating Features’, *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2017-Octob, pp. 3037–3046, 2017, doi: 10.1109/ICCV.2017.328.

[123] Z. Chen, Y. Fu, K. Chen, and Y. G. Jiang, ‘Image block augmentation for one-shot learning’, *33rd AAAI Conference on Artificial Intelligence, AAAI 2019, 31st Innovative Applications of Artificial Intelligence Conference, IAAI 2019 and the 9th AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019*, vol. 1, pp. 3379–3386, 2019, doi: 10.1609/aaai.v33i01.33013379.

[124] J. Zhou, Y. Zheng, J. Tang, J. Li, and Z. Yang, ‘FlipDA: Effective and Robust Data Augmentation for Few-Shot Learning’, *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, vol. 1, pp. 8646–8665, 2022, doi: 10.18653/v1/2022.acl-long.592.

[125] Z. Chen, Y. Fu, Y. X. Wang, L. Ma, W. Liu, and M. Hebert, ‘Image deformation meta-networks for one-shot learning’, *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2019-June, no. c, pp. 8672–8681, 2019, doi: 10.1109/CVPR.2019.00888.

[126] A. A. Rusu *et al.*, ‘Meta-learning with latent embedding optimization’, *7th International Conference on Learning Representations, ICLR 2019*, pp. 1–17, 2019.

[127] A. Nichol and J. Schulman, ‘Reptile: a scalable metalearning algorithm’, *arXiv preprint arXiv:1803.02999*, vol. 2, no. 3, p. 4, 2018.

[128] S. Ravi and H. Larochelle, ‘Optimization as a model for few-shot learning’, *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, pp. 1–11, 2017.

[129] C. Finn, P. Abbeel, and S. Levine, ‘Model-agnostic meta-learning for fast adaptation of deep networks’, *34th International Conference on Machine Learning, ICML 2017*, vol. 3, pp. 1856–1868, 2017.

[130] S. Chopra, R. Hadsell, and Y. LeCun, ‘Learning a similarity metric discriminatively, with application to face verification’, *Proceedings - 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2005*, vol. I, pp. 539–546, 2005, doi: 10.1109/CVPR.2005.202.

[131] S. Laenen and L. Bertinetto, ‘On Episodes, Prototypical Networks, and Few-Shot Learning’, *Adv Neural Inf Process Syst*, vol. 29, no. NeurIPS, pp. 24581–24592, 2021.

[132] G. S. Dhillon, P. Chaudhari, A. Ravichandran, and S. Soatto, ‘A Baseline for Few-Shot Image Classification’, pp. 1–20, 2019, [Online]. Available: http://arxiv.org/abs/1909.02729

[133] J. Snell, K. Swersky, and R. Zemel, ‘Prototypical networks for few-shot learning’, in *Advances in Neural Information Processing Systems*, 2017.

[134] Y. Chen, Z. Liu, H. Xu, T. Darrell, and X. Wang, ‘Meta-Baseline: Exploring Simple Meta-Learning for Few-Shot Learning’, *Proceedings of the IEEE International Conference on Computer Vision*, pp. 9042–9051, 2021, doi:

10.1109/ICCV48922.2021.00893.

[135] S. Gidaris, N. Komodakis, P. Paristech, and N. Komodakis, ‘Dynamic few-shot visual learning without forgetting’, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4367–4375. [Online]. Available: http://openaccess.thecvf.com/content_cvpr_2018/papers/Gidaris_Dynamic_Few-Shot_Visual_CVPR_2018_paper.pdf

[136] W. Y. Chen, Y. C. F. Wang, Y. C. Liu, Z. Kira, and J. Bin Huang, ‘A closer look at few-shot classification’, *7th International Conference on Learning Representations, ICLR 2019*, no. 2018, pp. 1–17, 2019.

[137] F. Sung, Y. Yang, and L. Zhang, ‘Relation Network for Few-Shot Learning’, *Cvpr*, pp. 1199–1208, 2018.

[138] G. S. Dhillon, P. Chaudhari, A. Ravichandran, and S. Soatto, ‘A Baseline for Few-Shot Image Classification’, *8th International Conference on Learning Representations, ICLR 2020*, pp. 1–20, 2020.

[139] G. Cheng, C. Lang, and J. Han, ‘Holistic Prototype Activation for Few-Shot Segmentation’, *IEEE Trans Pattern Anal Mach Intell*, vol. 45, no. 4, pp. 4650–4666, 2022, doi: 10.1109/TPAMI.2022.3193587.

[140] S. Cho, S. Park, G. Cha, and T. Oh, ‘Development of image processing for crack detection on concrete structures through terrestrial laser scanning associated with the octree structure’, *Applied Sciences*, vol. 8, no. 12, p. 2373, 2018.

[141] P. Stałowska, C. Suchocki, and M. Rutkowska, ‘Crack detection in building walls based on geometric and radiometric point cloud information’, *Autom Constr*, vol. 134, no. November 2021, 2022, doi: 10.1016/j.autcon.2021.104065.

[142] Q. Kong, J. Gu, B. Xiong, and C. Yuan, ‘Vision-aided three-dimensional damage quantification and finite element model geometric updating for reinforced concrete structures’, *Computer-Aided Civil and Infrastructure Engineering*, 2023, doi: 10.1111/mice.12967.

[143] Z. Zhang, ‘A flexible new technique for camera calibration’, *IEEE Trans Pattern Anal Mach Intell*, vol. 22, no. 11, pp. 1330–1334, 2000, doi: 10.1109/34.888718.

[144] H. Zhang, Y. Zou, E. del Rey Castillo, and X. Yang, ‘Detection of RC Spalling Damage and Quantification of Its Key Properties from 3D Point Cloud’, *KSCE Journal of Civil Engineering*, vol. 26, no. 5, pp. 2023–2035, 2022, doi: 10.1007/s12205-022-0890-y.

[145] C. Liu, L. Zhou, W. Wang, and X. Zhao, ‘Concrete Surface Damage Volume Measurement Based on Three-Dimensional Reconstruction by Smartphones’, *IEEE Sens J*, vol. 21, no. 10, pp. 11349–11360, 2021, doi: 10.1109/JSEN.2021.3067739.

[146] C. Tu, E. Takeuchi, A. Carballo, and K. Takeda, ‘Point cloud compression for 3d lidar

sensor using recurrent neural network with residual blocks', *Proc IEEE Int Conf Robot Autom*, vol. 2019-May, pp. 3274–3280, 2019, doi: 10.1109/ICRA.2019.8794264.

[147] T. Huang and Y. Liu, '3D point cloud geometry compression on deep learning', *MM 2019 - Proceedings of the 27th ACM International Conference on Multimedia*, pp. 890–898, 2019, doi: 10.1145/3343031.3351061.

[148] E. Aldemir, G. Tohumoglu, and M. A. Selver, 'Binary medical image compression using the volumetric run-length approach', *Imaging Science Journal*, vol. 67, no. 3, pp. 123–135, 2019, doi: 10.1080/13682199.2019.1565695.

[149] M. Artus, M. S. H. Alabassy, and C. Koch, 'A BIM Based Framework for Damage Segmentation, Modeling, and Visualization Using IFC', *Applied Sciences (Switzerland)*, vol. 12, no. 6, pp. 1–26, 2022, doi: 10.3390/app12062772.

[150] M. Artus and C. Koch, 'Object-Oriented Damage Information Modeling Concepts and Implementation for Bridge Inspection', *Journal of Computing in Civil Engineering*, vol. 36, no. 6, pp. 1–21, 2022, doi: 10.1061/(asce)cp.1943-5487.0001030.

[151] M. Artus and C. Koch, 'Modeling geometry and semantics of physical damages using IFC', *EG-ICE 2020 Workshop on Intelligent Computing in Engineering, Proceedings*, no. July, pp. 144–153, 2020.

[152] D. Isailović, V. Stojanovic, M. Trapp, R. Richter, R. Hajdin, and J. Döllner, 'Bridge damage: Detection, IFC-based semantic enrichment and visualization', *Autom Constr*, vol. 112, no. May 2019, p. 103088, 2020, doi: 10.1016/j.autcon.2020.103088.

[153] Y. Lei, B. Yang, X. Jiang, F. Jia, N. Li, and A. K. Nandi, 'Applications of machine learning to machine fault diagnosis: A review and roadmap', *Mech Syst Signal Process*, vol. 138, p. 106587, 2020, doi: 10.1016/j.ymssp.2019.106587.

[154] D. H. Pandya, S. H. Upadhyay, and S. P. Harsha, 'Fault diagnosis of rolling element bearing with intrinsic mode function of acoustic emission data using APF-KNN', *Expert Syst Appl*, vol. 40, no. 10, pp. 4137–4145, 2013, doi: 10.1016/j.eswa.2013.01.033.

[155] J. Wang, S. Liu, R. X. Gao, and R. Yan, 'Current envelope analysis for defect identification and diagnosis in induction motors', *J Manuf Syst*, vol. 31, no. 4, pp. 380–387, 2012, doi: 10.1016/j.jmsy.2012.06.005.

[156] X. Jiang, S. Li, and Y. Wang, 'A novel method for self-adaptive feature extraction using scaling crossover characteristics of signals and combining with LS-SVM for multi-fault diagnosis of gearbox', *Journal of Vibroengineering*, vol. 17, no. 4, pp. 1861–1878, 2015.

[157] T. Praveenkumar, B. Sabhrish, M. Saimurugan, and K. I. Ramachandran, 'Pattern recognition based on-line vibration monitoring system for fault diagnosis of automobile gearbox', *Measurement (Lond)*, vol. 114, no. April 2017, pp. 233–242, 2018, doi: 10.1016/j.measurement.2017.09.041.

[158] Z. Wang, Q. Zhang, J. Xiong, M. Xiao, G. Sun, and J. He, ‘Fault Diagnosis of a Rolling Bearing Using Wavelet Packet Denoising and Random Forests’, *IEEE Sens J*, vol. 17, no. 17, pp. 5581–5588, 2017, doi: 10.1109/JSEN.2017.2726011.

[159] H. Liu, L. Li, and J. Ma, ‘Rolling Bearing Fault Diagnosis Based on STFT-Deep Learning and Sound Signals’, *Shock and Vibration*, vol. 2016, 2016, doi: 10.1155/2016/6127479.

[160] C. Lu, Z. Y. Wang, W. L. Qin, and J. Ma, ‘Fault diagnosis of rotary machinery components using a stacked denoising autoencoder-based health state identification’, *Signal Processing*, vol. 130, pp. 377–388, 2017, doi: 10.1016/j.sigpro.2016.07.028.

[161] S. Qiu *et al.*, ‘Deep Learning Techniques in Intelligent Fault Diagnosis and Prognosis for Industrial Systems: A Review’, *Sensors*, vol. 23, no. 3, 2023, doi: 10.3390/s23031305.

[162] J. Ling, G. J. Liu, J. L. Li, X. C. Shen, and D. D. You, ‘Fault prediction method for nuclear power machinery based on Bayesian PPCA recurrent neural network model’, *Nuclear Science and Techniques*, vol. 31, no. 8, pp. 1–11, 2020, doi: 10.1007/s41365-020-00792-9.

[163] M. Yuan, Y. Wu, and L. Lin, ‘Fault diagnosis and remaining useful life estimation of aero engine using LSTM neural network’, *AUS 2016 - 2016 IEEE/CSAA International Conference on Aircraft Utility Systems*, pp. 135–140, 2016, doi: 10.1109/AUS.2016.7748035.

[164] A. C. Neves, I. González, and R. Karoumi, ‘A combined model-free Artificial Neural Network-based method with clustering for novelty detection: The case study of the KW51 railway bridge’, *IABSE Conference, Seoul 2020: Risk Intelligence of Infrastructures - Report*, pp. 181–188, 2021, doi: 10.2749/seoul.2020.181.

[165] A. C. Neves, I. González, J. Leander, and R. Karoumi, ‘Structural health monitoring of bridges: a model-free ANN-based approach to damage detection’, *J Civ Struct Health Monit*, vol. 7, no. 5, pp. 689–702, 2017, doi: 10.1007/s13349-017-0252-5.

[166] S. O. Sajedi and X. Liang, ‘Vibration-based semantic damage segmentation for large-scale structural health monitoring’, *Computer-Aided Civil and Infrastructure Engineering*, vol. 35, no. 6, pp. 579–596, 2020, doi: 10.1111/mice.12523.

[167] X. Wu, Z. Peng, J. Ren, C. Cheng, W. Zhang, and D. Wang, ‘Rub-Impact Fault Diagnosis of Rotating Machinery Based on 1-D Convolutional Neural Networks’, *IEEE Sens J*, vol. 20, no. 15, pp. 8349–8363, 2020, doi: 10.1109/JSEN.2019.2944157.

[168] S. Sony, S. Gamage, A. Sadhu, and J. Samarabandu, ‘Multiclass damage identification in a full-scale bridge using optimally tuned one-dimensional convolutional neural network’, *Journal of Computing in Civil Engineering*, vol. 36, no. 2, p. 4021035, 2022.

[169] S. Sharma and S. Sen, ‘One-dimensional convolutional neural network-based damage

detection in structural joints', *J Civ Struct Health Monit*, vol. 10, no. 5, pp. 1057–1072, 2020, doi: 10.1007/s13349-020-00434-z.

[170] Y. Zhang, Y. Miyamori, S. Mikami, and T. Saito, 'Vibration-based structural state identification by a 1-dimensional convolutional neural network', *Computer-Aided Civil and Infrastructure Engineering*, vol. 34, no. 9, pp. 822–839, 2019, doi: 10.1111/mice.12447.

[171] S. R. Fahim *et al.*, 'A Robust Self-Attentive Capsule Network for Fault Diagnosis of Series-Compensated Transmission Line', *IEEE Transactions on Power Delivery*, vol. 36, no. 6, pp. 3846–3857, 2021, doi: 10.1109/TPWRD.2021.3049861.

[172] J. Jiang *et al.*, 'Fault diagnosis of the bushing infrared images based on mask R-CNN and improved PCNN joint algorithm', *High Voltage*, vol. 6, no. 1, pp. 116–124, 2021, doi: 10.1049/hve.2019.0249.

[173] M. Xia, T. Li, L. Xu, L. Liu, and C. W. De Silva, 'Fault Diagnosis for Rotating Machinery Using Multiple Sensors and Convolutional Neural Networks', *IEEE/ASME Transactions on Mechatronics*, vol. 23, no. 1, pp. 101–110, 2018, doi: 10.1109/TMECH.2017.2728371.

[174] L. Gou, H. Li, H. Zheng, H. Li, and X. Pei, 'Aeroengine Control System Sensor Fault Diagnosis Based on CWT and CNN', *Math Probl Eng*, vol. 2020, 2020, doi: 10.1155/2020/5357146.

[175] S. Meng, J. Kang, K. Chi, and X. Die, 'Intelligent fault diagnosis of gearbox based on multiple synchrosqueezing S-transform and convolutional neural networks', *International Journal of Performability Engineering*, vol. 16, no. 4, pp. 528–536, 2020, doi: 10.23940/ijpe.20.04.p4.528536.

[176] D. T. Hoang and H. J. Kang, 'Rolling element bearing fault diagnosis using convolutional neural network and vibration image', *Cogn Syst Res*, vol. 53, pp. 42–50, 2019, doi: 10.1016/j.cogsys.2018.03.002.

[177] S. Wang, J. Xiang, Y. Zhong, and Y. Zhou, 'Convolutional neural network-based hidden Markov models for rolling element bearing fault identification', *Knowl Based Syst*, vol. 144, pp. 65–76, 2018, doi: 10.1016/j.knosys.2017.12.027.

[178] L. Wan, Y. Chen, H. Li, and C. Li, 'Rolling-element bearing fault diagnosis using improved lenet-5 network', *Sensors (Switzerland)*, vol. 20, no. 6, pp. 1–23, 2020, doi: 10.3390/s20061693.

[179] S. Teng, G. Chen, Z. Liu, L. Cheng, and X. Sun, 'Multi-sensor and decision-level fusion-based structural damage detection using a one-dimensional convolutional neural network', *Sensors*, vol. 21, no. 12, 2021, doi: 10.3390/s21123950.

[180] Q. Gong, W., Chen, H., Zhang, Z., Zhang, M., Wang, R., Guan, C., & Wang, 'A Novel Deep Learning Method for Intelligent Fault Diagnosis of Rotating Machinery Based on

Improved CNN-SVM and Multichannel Data Fusion', 2019.

[181] 'Knowledge Engineering: What it Means, Examples'. Accessed: Aug. 09, 2023. [Online]. Available: https://www.investopedia.com/terms/k/knowledge-engineering.asp

[182] Y. Zhang, J. Liu, and K. Hou, 'Building a Knowledge Base of Bridge Maintenance Using Knowledge Graph', vol. 2023, 2023.

[183] G. Ren, R. Ding, and H. Li, 'Building an ontological knowledgebase for bridge maintenance', *Advances in Engineering Software*, vol. 130, no. July 2018, pp. 24–40, 2019, doi: 10.1016/j.advengsoft.2019.02.001.

[184] K. Liu and N. El-Gohary, 'Bridge Deterioration Knowledge Ontology for Supporting Bridge Document Analytics', *J Constr Eng Manag*, vol. 148, no. 6, pp. 1–14, 2022, doi: 10.1061/(asce)co.1943-7862.0002210.

[185] C. Wu, P. Wu, J. Wang, R. Jiang, M. Chen, and X. Wang, 'Ontological knowledge base for concrete bridge rehabilitation project management', *Autom Constr*, vol. 121, no. May 2020, p. 103428, 2021, doi: 10.1016/j.autcon.2020.103428.

[186] R. Li, T. Mo, J. Yang, S. Jiang, T. Li, and Y. Liu, 'Ontologies-Based Domain Knowledge Modeling and Heterogeneous Sensor Data Integration for Bridge Health Monitoring Systems', *IEEE Trans Industr Inform*, vol. 17, no. 1, pp. 321–332, 2021, doi: 10.1109/TII.2020.2967561.

[187] J. Yang *et al.*, 'Intelligent bridge management via big data knowledge engineering', *Autom Constr*, vol. 135, no. January, p. 104118, 2022, doi: 10.1016/j.autcon.2021.104118.

[188] R. Li, T. Mo, J. Yang, D. Li, S. Jiang, and D. Wang, 'Bridge inspection named entity recognition via BERT and lexicon augmented machine reading comprehension neural model', *Advanced Engineering Informatics*, vol. 50, no. September, p. 101416, 2021, doi: 10.1016/j.aei.2021.101416.

[189] K. Liu and N. El-Gohary, 'Ontology-based semi-supervised conditional random fields for automated information extraction from bridge inspection reports', *Autom Constr*, vol. 81, pp. 313–327, 2017, doi: 10.1016/j.autcon.2017.02.003.

[190] K. Liu and N. El-Gohary, 'Semantic Neural Network Ensemble for Automated Dependency Relation Extraction from Bridge Inspection Reports', *Journal of Computing in Civil Engineering*, vol. 35, no. 4, 2021, doi: 10.1061/(asce)cp.1943-5487.0000961.

[191] R. Li *et al.*, 'Joint extraction of entities and relations via an entity correlated attention neural model', *Inf Sci (N Y)*, vol. 581, pp. 179–193, 2021, doi: 10.1016/j.ins.2021.09.028.

[192] B. Sanchez-Lengeling, E. Reif, A. Pearce, and A. B. Wiltschko, 'A Gentle Introduction to Graph Neural Networks', *Distill*, vol. 6, no. 9, p. e33, Sep. 2021, doi:

10.23915/DISTILL.00033.

[193] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, ‘A Comprehensive Survey on Graph Neural Networks’, *IEEE Trans Neural Netw Learn Syst*, vol. 32, no. 1, pp. 4–24, 2021, doi: 10.1109/TNNLS.2020.2978386.

[194] J. Zhou *et al.*, ‘Graph neural networks: A review of methods and applications’, *AI Open*, vol. 1, no. April, pp. 57–81, 2020, doi: 10.1016/j.aiopen.2021.01.001.

[195] F. C. Collins, A. Braun, M. Ringsquandl, D. M. Hall, and A. Borrmann, ‘Assessing IFC classes with means of geometric deep learning on different graph encodings’, *Proceedings of the 2021 European Conference on Computing in Construction*, vol. 2, pp. 332–341, 2021, doi: 10.35490/ec3.2021.168.

[196] Z. Wang, R. Sacks, and T. Yeung, ‘Exploring graph neural networks for semantic enrichment: Room type classification’, *Autom Constr*, vol. 134, no. December 2021, p. 104039, 2022, doi: 10.1016/j.autcon.2021.104039.

[197] L. Xia, Y. Liang, P. Zheng, and J. Leng, ‘Maintenance planning recommendation of complex industrial equipment based on knowledge graph and graph neural network’, *Reliab Eng Syst Saf*, vol. 232, no. December 2022, p. 109068, 2023, doi: 10.1016/j.ress.2022.109068.

[198] M. N. K. Saunders, P. Lewis, and A. Thornhill, *Chapter 4: Understanding research philosophy and approaches to theory development*, no. March. 2019. [Online]. Available: https://www.researchgate.net/publication/330760964_Research_Methods_for_Business_Students_Chapter_4_Understanding_research_philosophy_and_approaches_to_theory_development

[199] D. Deb, R. Dey, and V. E. Balas, *Introduction: What is research?*, vol. 153. 2019. doi: 10.1007/978-981-13-2947-0_1.

[200] ‘Guide to Understanding the Research Onion | 15 Writers’. Accessed: Sep. 20, 2023. [Online]. Available: https://15writers.com/research-onion/

[201] K. Nallaperumal, ‘Engineering Research Methodology Krishnan Nallaperumal First Edition’, 2004.

[202] N. Mackenzie and S. Knipe, ‘Research dilemmas: Paradigms, methods and methodology.’, *Issues in educational research*, vol. 16, no. 2, pp. 193–205, 2006.

[203] R. K. Verma, K. K. Pattanaik, P. B. R. Dissanayake, A. J. Dammika, H. A. D. Buddika, and M. R. Kaloop, ‘Damage detection in bridge structures: An edge computing approach’, *arXiv preprint arXiv:2008.06724*, 2020.

[204] A. Anaissi, B. Suleiman, and W. Alyassine, ‘Personalised federated learning framework for damage detection in structural health monitoring’, *J Civ Struct Health Monit*, 2022, doi: 10.1007/s13349-022-00615-y.

[205] 'Artificial intelligence of things - Wikipedia'. Accessed: Jan. 28, 2023. [Online]. Available: https://en.wikipedia.org/wiki/Artificial_intelligence_of_things

[206] D. Harris-birtill and R. Harris-birtill, 'Understanding Computation Time: A Critical Discussion of Time as a Computational Performance Metric', *Time in Variance*, vol. 2, pp. 220–248, 2021, doi: 10.1163/9789004470170_014.

[207] W. Dean, 'Computational complexity theory', 2015.

[208] S. Baruah and A. Burns, 'Sustainable scheduling analysis', *Proceedings - Real-Time Systems Symposium*, pp. 159–168, 2006, doi: 10.1109/RTSS.2006.47.

[209] S. Baškarada and A. Koronios, 'Data, information, knowledge, wisdom (DIKW): A semiotic theoretical and empirical exploration of the hierarchy and its quality dimension', *Australasian Journal of Information Systems*, vol. 18, no. 1, pp. 5–24, 2013, doi: 10.3127/ajis.v18i1.748.

[210] E. Bianchi, A. L. Abbott, P. Tokekar, and M. Hebdon, 'COCO-Bridge: Structural Detail Data Set for Bridge Inspections', *Journal of Computing in Civil Engineering*, vol. 35, no. 3, 2021, doi: 10.1061/(asce)cp.1943-5487.0000949.

[211] A. A. Razborov, 'Communication Complexity', *An Invitation to Mathematics*, pp. 97–117, 2011, doi: 10.1007/978-3-642-19533-4_8.

[212] 'Fault tolerance - Wikipedia'. Accessed: Aug. 27, 2022. [Online]. Available: https://en.wikipedia.org/wiki/Fault_tolerance

[213] M. Chiachío, M. Megía, J. Chiachío, J. Fernandez, and M. L. Jalón, 'Structural digital twin framework: Formulation and technology integration', *Autom Constr*, vol. 140, no. May, p. 104333, 2022, doi: 10.1016/j.autcon.2022.104333.

[214] T. Murata, 'Petri Nets: Properties, Analysis and Applications', *Proceedings of the IEEE*, vol. 77, no. 4. pp. 541–580, 1989. doi: 10.1109/5.24143.

[215] J. R. Silva and P. M. G. Foyo, 'Timed petri nets', *Computer Networks and ISDN Systems*, vol. 10, no. 5, pp. 312–313, 1985, doi: 10.1016/0169-7552(85)90073-X.

[216] W. Li *et al.*, 'On Enabling Sustainable Edge Computing with Renewable Energy Resources', *IEEE Communications Magazine*, vol. 56, no. 5, pp. 94–101, 2018, doi: 10.1109/MCOM.2018.1700888.

[217] K. Mekki, E. Bajic, F. Chaxel, and F. Meyer, 'Overview of Cellular LPWAN Technologies for IoT Deployment: Sigfox, LoRaWAN, and NB-IoT', *2018 IEEE International Conference on Pervasive Computing and Communications Workshops, PerCom Workshops 2018*, pp. 197–202, 2018, doi: 10.1109/PERCOMW.2018.8480255.

[218] V. K. Sarker, J. P. Queralta, T. N. Gia, H. Tenhunen, and T. Westerlund, 'A survey on LoRa for IoT: Integrating edge computing', *2019 4th International Conference on Fog and Mobile Edge Computing, FMEC 2019*, no. June, pp. 295–300, 2019, doi:

10.1109/FMEC.2019.8795313.

[219] C. S. Shim, N. S. Dang, S. Lon, and C. H. Jeon, ‘Development of a bridge maintenance system for prestressed concrete bridges using 3D digital twin model’, *Structure and Infrastructure Engineering*, vol. 15, no. 10, pp. 1319–1332, Oct. 2019, doi: 10.1080/15732479.2019.1620789.

[220] H. Xu, X. Su, Y. Wang, H. Cai, K. Cui, and X. Chen, ‘Automatic bridge crack detection using a convolutional neural network’, *Applied Sciences (Switzerland)*, vol. 9, no. 14, 2019, doi: 10.3390/app9142867.

[221] D. Tan, ‘Assessment of Existing Bridge’s Beam Bending Stiffness Using Crack Characteristics’, *Engineering*, vol. 12, no. 02, pp. 82–89, 2020, doi: 10.4236/eng.2020.122008.

[222] T. Chen, D. Eager, and D. Makaroff, ‘Efficient image transmission using lora technology in agricultural monitoring iot systems’, in *2019 International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, IEEE, 2019, pp. 937–944. doi: 10.1109/iThings/GreenCom/CPSCom/SmartData.2019.00166.

[223] C. Pham, ‘Robust CSMA for long-range LoRa transmissions with image sensing devices’, *IFIP Wireless Days*, vol. 2018-April, pp. 116–122, 2018, doi: 10.1109/WD.2018.8361706.

[224] ‘Signal Features - MATLAB & Simulink - MathWorks United Kingdom’. Accessed: Feb. 01, 2023. [Online]. Available: https://uk.mathworks.com/help/predmaint/ug/signal-features.html

[225] J. Wang, Y. Sun, Z. Liu, P. Yang, and T. Lin, ‘Route planning based on floyd algorithm for intelligence transportation system’, *IEEE ICIT 2007 - 2007 IEEE International Conference on Integration Technology*, pp. 544–546, 2007, doi: 10.1109/ICITECHNOLOGY.2007.4290376.

[226] D. K. Fan and P. Shi, ‘Improvement of Dijkstraa’s algorithm and its application in route planning’, *Proceedings - 2010 7th International Conference on Fuzzy Systems and Knowledge Discovery, FSKD 2010*, vol. 4, no. Fskd, pp. 1901–1904, 2010, doi: 10.1109/FSKD.2010.5569452.

[227] B. al Homssi, K. Dakic, S. Maselli, H. Wolf, S. Kandeepan, and A. Al-Hourani, ‘IoT Network Design Using Open-Source LoRa Coverage Emulator’, *IEEE Access*, vol. 9, pp. 53636–53646, 2021, doi: 10.1109/ACCESS.2021.3070976.

[228] S. Jeong, R. Hou, J. P. Lynch, H. Sohn, and K. H. Law, ‘A scalable cloud-based cyberinfrastructure platform for bridge monitoring’, *Structure and Infrastructure Engineering*, vol. 15, no. 1, pp. 82–102, Jan. 2019, doi:

10.1080/15732479.2018.1500617.

[229] T. Mitchell, *Machine Learning*. McGraw-Hill Education, 1997.

[230] Y. Chen, T. Darrell, and X. Wang, 'Meta-Baseline : Exploring Simple Meta-Learning for Few-Shot Learning', pp. 9062–9071.

[231] Y. Wang, Q. Yao, J. T. Kwok, and L. M. Ni, 'Generalizing from a Few Examples: A Survey on Few-shot Learning', *ACM Comput Surv*, vol. 53, no. 3, 2020, doi: 10.1145/3386252.

[232] C. Finn, P. Abbeel, and S. Levine, 'Model-agnostic meta-learning for fast adaptation of deep networks', *34th International Conference on Machine Learning, ICML 2017*, vol. 3, pp. 1856–1868, 2017.

[233] A. A. Rusu *et al.*, 'Meta-learning with latent embedding optimization', *7th International Conference on Learning Representations, ICLR 2019*, pp. 1–17, 2019.

[234] O. Vinyals, C. Blundell, T. Lillicrap, K. Kavukcuoglu, and D. Wierstra, 'Matching networks for one shot learning', *Adv Neural Inf Process Syst*, no. Nips, pp. 3637–3645, 2016.

[235] M. Devgan, G. Malik, and D. K. Sharma, *Semi-Supervised Learning*. 2020. doi: 10.1002/9781119654834.ch10.

[236] S. Panigrahi, A. Nanda, and T. Swarnkar, 'A Survey on Transfer Learning', *Smart Innovation, Systems and Technologies*, vol. 194, no. 10, pp. 781–789, 2021, doi: 10.1007/978-981-15-5971-6_83.

[237] Q. Sun and Y. L. T. Chua, 'Meta-Transfer Learning for Few-Shot Learning', pp. 403–412, 2019.

[238] K. He, X. Chen, S. Xie, Y. Li, P. Dollar, and R. Girshick, 'Masked Autoencoders Are Scalable Vision Learners', pp. 15979–15988, 2022, doi: 10.1109/cvpr52688.2022.01553.

[239] H. Bao, L. Dong, S. Piao, and F. Wei, 'BEiT: BERT Pre-Training of Image Transformers', no. Mim, pp. 1–18, 2021.

[240] Y. Chen *et al.*, 'Meta-Baseline: Exploring Simple Meta-Learning for Few-Shot Learning', *Proceedings of the IEEE International Conference on Computer Vision*, pp. 9042–9051, 2021, doi: 10.1109/ICCV48922.2021.00893.

[241] W. Y. Chen, Y. C. F. Wang, Y. C. Liu, Z. Kira, and J. Bin Huang, 'A closer look at few-shot classification', *7th International Conference on Learning Representations, ICLR 2019*, no. 2018, pp. 1–17, 2019.

[242] M. Mundt, S. Majumder, S. Murali, P. Panetsos, and V. Ramesh, 'Meta-learning convolutional neural architectures for multi-target concrete defect classification with the

concrete defect bridge image dataset', *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2019-June, pp. 11188–11197, 2019, doi: 10.1109/CVPR.2019.01145.

[243] 'Fit plane to 3-D point cloud - MATLAB pcfitplane - MathWorks United Kingdom'. Accessed: Aug. 02, 2023. [Online]. Available: https://uk.mathworks.com/help/vision/ref/pcfitplane.html

[244] 'Random sample consensus - Wikipedia'. Accessed: Aug. 02, 2023. [Online]. Available: https://en.wikipedia.org/wiki/Random_sample_consensus

[245] 'Your garden walls : better to be safe', *Garden*, pp. 4–6, Accessed: May 29, 2023. [Online]. Available: https://www.gov.uk/guidance/your-garden-walls-better-to-be-safe

[246] D. buildings Ltd, 'Cracking and building movement', *Structural Survey*, vol. 23, no. 1, Feb. 2005, doi: 10.1108/ss.2005.11023aae.007.

[247] 'NR-L3-CIV-006 - Handbook For The Examination of Structures | PDF | Highway | Road'. Accessed: Jun. 15, 2023. [Online]. Available: https://www.scribd.com/document/632406023/NR-L3-CIV-006-Handbook-for-the-Examination-of-Structures

[248] E. Bianchi and M. Hebdon, 'Concrete Crack Conglomerate Dataset', *University Libraries*. 2021. Accessed: May 29, 2023. [Online]. Available: https://data.lib.vt.edu/articles/dataset/Concrete_Crack_Conglomerate_Dataset/16625056/1

[249] A. Akagic, E. Buza, S. Omanovic, and A. Karabegovic, 'Pavement crack detection using Otsu thresholding for image segmentation', *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics, MIPRO 2018 - Proceedings*, pp. 1092–1097, 2018, doi: 10.23919/MIPRO.2018.8400199.

[250] M. Everingha, L. van Gool, C. Williams, J. Winn, and A. Zisserman, 'The PASCAL Visual Object Classes Homepage', *Pascal-Voc*. 2018. Accessed: May 29, 2023. [Online]. Available: http://host.robots.ox.ac.uk/pascal/VOC/

[251] Y. Xu, X. Tong, and U. Stilla, 'Voxel-based representation of 3D point clouds: Methods, applications, and its potential use in the construction industry', *Autom Constr*, vol. 126, no. February, p. 103675, 2021, doi: 10.1016/j.autcon.2021.103675.

[252] M. Artus, M. S. H. Alabassy, and C. Koch, 'A BIM Based Framework for Damage Segmentation, Modeling, and Visualization Using IFC', *Applied Sciences (Switzerland)*, vol. 12, no. 6, pp. 10–12, 2022, doi: 10.3390/app12062772.

[253] C. Bauhaus-universit, 'IFC based Framework for Generating , Modeling and Visualizing Spalling Defect Geometries', pp. 1–11, 2018.

[254] P. Stałowska and C. Suchocki, 'TLS data for cracks detection in building walls', *Data*

Brief, vol. 42, no. May, p. 108247, 2022, doi: 10.1016/j.dib.2022.108247.

[255] 'Automated machine learning - Wikipedia'. Accessed: Aug. 16, 2023. [Online]. Available: https://en.wikipedia.org/wiki/Automated_machine_learning

[256] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, 'Hyperband: A novel bandit-based approach to hyperparameter optimization', *Journal of Machine Learning Research*, vol. 18, pp. 1–52, 2018.

[257] J. Wu, X. Y. Chen, H. Zhang, L. D. Xiong, H. Lei, and S. H. Deng, 'Hyperparameter optimization for machine learning models based on Bayesian optimization', *Journal of Electronic Science and Technology*, vol. 17, no. 1, pp. 26–40, 2019, doi: 10.11989/JEST.1674-862X.80904120.

[258] Y. Ozaki, Y. Tanigaki, S. Watanabe, and M. Onishi, 'Multiobjective tree-structured parzen estimator for computationally expensive optimization problems', *GECCO 2020 - Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, pp. 533–541, 2020, doi: 10.1145/3377930.3389817.

[259] A. Li *et al.*, 'A generalized framework for population based training', *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1791–1799, 2019, doi: 10.1145/3292500.3330649.

[260] Y. Zhang, R. Jin, and Z. H. Zhou, 'Understanding bag-of-words model: A statistical framework', *International Journal of Machine Learning and Cybernetics*, vol. 1, no. 1–4, pp. 43–52, 2010, doi: 10.1007/s13042-010-0001-0.

[261] S. Qaiser and R. Ali, 'Text Mining: Use of TF-IDF to Examine the Relevance of Words to Documents', *Int J Comput Appl*, vol. 181, no. 1, pp. 25–29, 2018, doi: 10.5120/ijca2018917395.

[262] K. W. Church, 'Emerging Trends: Word2Vec', *Nat Lang Eng*, vol. 23, no. 1, pp. 155–162, 2017, doi: 10.1017/S1351324916000334.

[263] J. Pennington, R. Socher, and C. D. Manning, 'Glove: Global vectors for word representation', in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[264] J. Devlin, M.-W. Chang, K. Lee, K. T. Google, and A. I. Language, 'BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding', *Naacl-Hlt 2019*, no. Mlm, pp. 4171–4186, 2018.

[265] B. Min *et al.*, 'Recent Advances in Natural Language Processing via Large Pre-Trained Language Models: A Survey', *ACM Comput Surv*, 2023, doi: 10.1145/3605943.

[266] H. Wang, J. Li, H. Wu, E. Hovy, and Y. Sun, 'Pre-Trained Language Models and Their Applications', *Engineering*, no. xxxx, 2022, doi: 10.1016/j.eng.2022.04.024.

[267] T. Kim, K. M. Yoo, and S. G. Lee, 'Self-guided contrastive learning for BERT sentence

representations', *ACL-IJCNLP 2021 - 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, Proceedings of the Conference*, pp. 2528–2540, 2021, doi: 10.18653/v1/2021.acl-long.197.

[268] S. Zhang, H. Tong, J. Xu, and R. Maciejewski, 'Graph convolutional networks: a comprehensive review', *Comput Soc Netw*, vol. 6, no. 1, 2019, doi: 10.1186/s40649-019-0069-y.

[269] W. L. Hamilton, R. Ying, and J. Leskovec, 'Inductive representation learning on large graphs', *Adv Neural Inf Process Syst*, vol. 2017-Decem, no. Nips, pp. 1025–1035, 2017.

[270] Z. Wang, H. Ying, R. Sacks, and A. Borrmann, 'CBIM : A Graph-based Approach to Enhance Interoperability Using Semantic Enrichment', pp. 1–10, 2022.

[271] Z. Wang, R. Sacks, and T. Yeung, 'Exploring graph neural networks for semantic enrichment: Room type classification', *Autom Constr*, vol. 134, no. October 2021, p. 104039, 2022, doi: 10.1016/j.autcon.2021.104039.

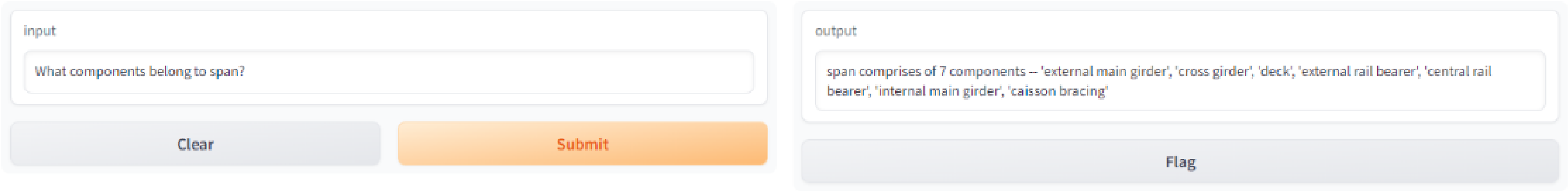
[272] 'Link Prediction using Graph Neural Networks — DGL 1.1.1 documentation'. Accessed: Aug. 09, 2023. [Online]. Available: https://docs.dgl.ai/tutorials/blitz/4_link_predict.html#sphx-glr-tutorials-blitz-4-link-predict-py

Appendix A: Bridge inspection and repair actions – 15092 (primary data)

Main	Component	Element	Defect	Action
span	external main girder	stiffener crank	remarkable section loss	cut out corroded section and weld a new plate in its place
span	external main girder	stiffener crank	medium section loss	cut out corroded steel to a specified thickness and apply a specified radius for the new coating system
span	external main girder	bottom flange	section loss	repair the half width of the bottom flange
span	cross girder	bottom flange	section loss	repair the full width of the bottom flange
span	external main girder	bottom flange	remarkable section loss	cut out corroded section and weld a new plate in its place
span	deck	plate	hole	weld a new plate to the underside of the steel deck
span	cross girder	bottom flange	section loss rivet head loss and thinning	repair the full width of the bottom flange
span	cross girder	fixing	failure	cut out fixing and ream the hole
span	cross girder	web plate	hole	install a packer plate between the upper and lower angles and a cover plate over the packer
span	external rail bearer	web plate	hole	install a welded or bolted plate past the defect area
span	central rail bearer	web plate	hole	install a welded or bolted plate past the defect area
span	deck	stiffener gusset plate	section loss	replace gusset plate
span	internal main girder	bottom flange	section loss	repair the half width of the bottom flange
span	internal main girder	stiffener crank	remarkable section loss	cut out corroded section and weld a new plate in its place

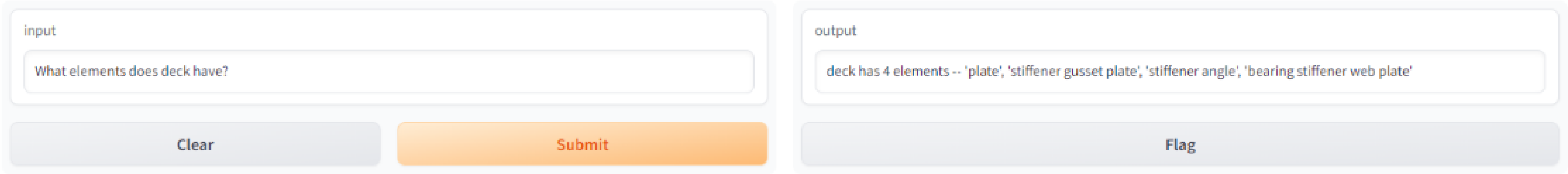
span	internal main girder	stiffener crank	medium section loss	cut out corroded steel to a specified thickness and apply a specified radius for the new coating system
span	central rail bearer	bottom flange	excessive section loss	replace the whole bottom flange with a tee section and bolted splice or replace a measured section with angles and flange splice
span	deck	stiffener angle	deformation	replace stiffener
span	external rail bearer	bottom flange	excessive section loss	replace the whole bottom flange with a tee section and bolted splice or replace a measured section with angles and flange splice
span	external main girder	web plate	hole and pitting	install a bolted packer plate to the main girder web and strengthen cover plate over the stiffener and lower angle
span	external main girder	stiffener crank	remarkable section loss	install a bolted packer plate to the main girder web and strengthen cover plate over the stiffener and lower angle
span	external main girder	stiffener crank	slight section loss	fill with similar product
span	central rail bearer	cleat connection	section loss and pitting	replace cleat
span	external main girder	bearing web plate	hole and pitting	install a bolted packer plate to the main girder web and splice it to the stiffener angles for both web sides
span	deck	bearing stiffener web plate	hole	remove existing rivets using burning equipment and install a new steel plate to the stiffener web
span	internal main girder	bearing stiffener crank	section loss	cut out corroded steel to a specified thickness and apply a specified radius for the new coating system
span	caisson bracing	top flange	section loss	replace the back with new
span	caisson bracing	web plate	hole	install a packer plate between the upper and lower angles and a cover plate over the packer

Appendix B: Templated chatbot for bridge maintenance query



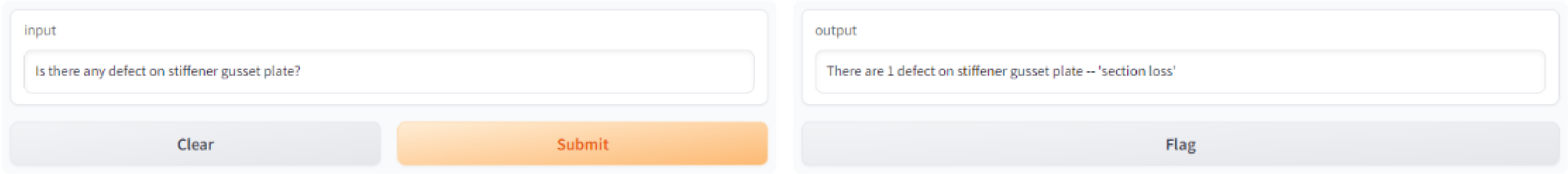
What components belong to span?

span comprises of 7 components -- 'external main girder', 'cross girder', 'deck', 'external rail bearer', 'central rail bearer', 'internal main girder', 'caisson bracing'



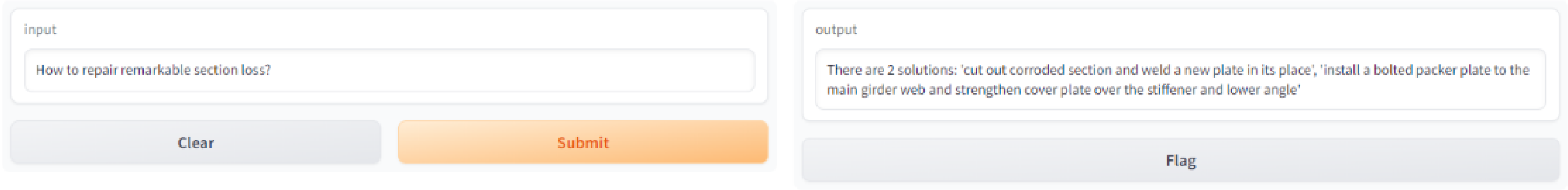
What elements does deck have?

deck has 4 elements -- 'plate', 'stiffener gusset plate', 'stiffener angle', 'bearing stiffener web plate'



Is there any defect on stiffener gusset plate?

There are 1 defect on stiffener gusset plate -- 'section loss'



How to repair remarkable section loss?

There are 2 solutions: 'cut out corroded section and weld a new plate in its place', 'install a bolted packer plate to the main girder web and strengthen cover plate over the stiffener and lower angle'