

論文 / 著書情報
Article / Book Information

題目(和文)	
Title(English)	Spatial Roughness Learning with Neural Networks for Collapsed Building Detection Using Post-Earthquake 3D Point Clouds
著者(和文)	修浩毅
Author(English)	Haoyi Xiu
出典(和文)	学位:博士(学術), 学位授与機関:東京工業大学, 報告番号:甲第12484号, 授与年月日:2023年3月26日, 学位の種別:課程博士, 審査員:松岡 昌志,山中 浩明,浅輪 貴史,藤田 康仁,大風 翼
Citation(English)	Degree:Doctor (Academic), Conferring organization: Tokyo Institute of Technology, Report number:甲第12484号, Conferred date:2023/3/26, Degree Type:Course doctor, Examiner:,,,,
学位種別(和文)	博士論文
Type(English)	Doctoral Thesis

TOKYO INSTITUTE OF TECHNOLOGY

Spatial Roughness Learning with Neural
Networks for Collapsed Building Detection Using
Post-Earthquake 3D Point Clouds

Author:

Haoyi XIU

Supervisor:

Prof. Masashi MATSUOKA

*A dissertation submitted in fulfillment of the requirements
for the degree of Doctor of Philosophy*

in the

School of Environment and Society
Department of Architecture and Building Engineering

February 15, 2023

Abstract

In this research, we deal with deep learning-based collapsed building detection (CBD) using 3D point clouds. With regard to this topic, two issues that hinder its further advancement are found through extensive literature review: 1) the lack of a suitable dataset on which the deep learning models can be trained and evaluated, and 2) the deep learning models are yet to be adapted to CBD. Therefore, this research aims to resolve the above issues to investigate the potential of deep learning in performing CBD.

To cope with the first issue, we create the Mashiki dataset, an airborne 3D point cloud dataset developed using the data acquired after the 2016 Kumamoto earthquake, and analyze the performance of mainstream deep learning methods in detecting collapsed buildings.

To resolve the second issue, we develop Laplacian-Enhanced neural network (LE-Net), a dedicated method for CBD. The core of LE-Net is Laplacian Unit (LU), a module inspired by the nature of building damage. LU is designed to enhance the useful discontinuities while suppressing the less useful ones.

Extensive experiments on the Mashiki dataset verify the effectiveness and design choices of LE-Net. Moreover, its decision-making is explored through visual explanation analysis. In addition, the computational efficiency, robustness of LE-Net, and applicability to various scenarios are also examined. Furthermore, the adaptability and transferability of LE-Net are validated using a significantly different dataset.

The dissertation concludes with a discussion of limitations and future work.

Synopsis

Collapsed buildings should be detected immediately after the occurrence of natural hazards for humanitarian assistance and post-disaster recovery. Airborne 3D point clouds are especially useful for this purpose as they precisely record the height of buildings. Recently, deep learning has been extremely successful in 3D point cloud processing. Despite its potential, the use of deep learning in collapsed building detection (CBD) using 3D point clouds is largely unexplored. Therefore, this research aims to explore the use of deep learning for CBD using 3D point clouds.

Chapter 1 formulates the objectives of this research. Extensive literature review and analysis show that there are two major problems that need to be solved. The first problem is the lack of an appropriate dataset. The lack of a dataset is a fundamental problem because deep learning requires a large-scale and well-prepared dataset. Furthermore, this problem also makes it impossible to investigate the potential of deep learning on CBD. The second problem is that the deep learning models are yet to be adapted to CBD. CBD is a specific application where clear assumptions can be made and implemented to improve the model design. The absence of such a design may significantly limit the performance. Accordingly, the objectives of this research are set: 1) the development of an appropriate dataset and 2) the development of a dedicated model for CBD.

Chapter 2 develops the dataset and performs benchmarking of some mainstream algorithms. The field survey and airborne LiDAR point clouds obtained after the 2016 Kumamoto earthquake are used as data sources to develop the dataset. The dataset is termed the Mashiki dataset. We design a detailed workflow consisting of five consecutive steps: 1) information extraction from the field survey; 2) building GPS coordinates adjustment; 3) building annotation; 4) visual interpretation of damage; and 5) filtering buildings. Next, we benchmark several mainstream algorithms on the Mashiki dataset. The performance of PointNet++, the standard model for point cloud analysis, is carefully investigated. The result reveals that PointNet++ fails to detect buildings that show very few damage cues.

Chapter 3 elaborates on the development of Laplacian-enhanced neural network (LE-Net), a dedicated method for CBD. The chapter begins with an analysis concerning damage characteristics and the numerical representation

of building damage. Based on the above analysis, the discrete Laplacian is chosen as the tool to encode building damage because it successfully represents various types of collapse patterns. However, it responds to all discontinuities including less useful ones equally, lacking in selectivity. To resolve this problem, we propose Laplacian Unit (LU), a module designed to enhance the useful discontinuities in preference to less useful ones. Specifically, such functionality is realized by the newly introduced selectivity function that transforms the discrete Laplacian in a trainable manner. As a result, we expect that the optimized Laplacian can extract enhanced spatial patterns of building damage so that challenging collapsed buildings can be detected. Using LU as a core, we construct a dedicated network called LE-Net to tackle CBD. LE-Net enhances PointNet++ by integrating LUs at different stages of PointNet++.

Chapter 4 performs comprehensive evaluations of LE-Net. In the first part, we perform diverse experiments on the Mashiki dataset. Quantitatively, LE-Net achieves the best overall performance and outperforms PointNet++ in terms of all performance metrics. Qualitatively, LE-Net successfully detects collapsed buildings that are completely missed by PointNet++, demonstrating its enhanced sensitivity to challenging buildings. Next, we validate the usefulness of each component of LU. In particular, it is verified that the selectivity function enhances collapse-related representations while smoothing out other unrelated discontinuities, meeting our expectations. Visual explanation analysis is subsequently performed to reveal the decision-making of LE-Net. The result indicates that LE-Net successfully locates various damage to buildings. Further, the computational efficiency and robustness of the density variation are analyzed. The applicability of LE-Net to a real-world scenario and ground deformation-related damage is verified. In the second part, we validate the adaptability and transferability of LE-Net on a substantially different dataset. For this purpose, the Haiti dataset is developed using data obtained after the 2010 Haiti earthquake. As a result, LE-Net shows better adaptability compared with PointNet++. Furthermore, LE-Net shows superior transferability by outperforming PointNet++ in various situations.

The dissertation concludes with a discussion of limitations and future work.

Acknowledgements

First of all, I would like to thank Professor Matsuoka for his guidance throughout the doctoral program. His continuous guidance on how to proceed logically with my research was extremely helpful in developing skills that I will need for the rest of my life. I would also like to thank him for giving me the freedom to choose what I wanted to do and for continuing to give me constructive advice whenever I needed it. Under his guidance, I not only obtained my degree and skills but also gained an understanding of what I should pursue throughout my research life.

It was my honor to have my research evaluated by Prof. Yamanaka, Asawa, Fujita, and Okaze. Their remarks added much significance and clarity to my research. Moreover, I thank them for asking me many questions that were designed not only to improve my research but also to improve the way I think.

I would also like to thank my lab members. Asst. Prof. Oshio gave me precise suggestions that always improve my research instantly. I also thank Dr. Ogushi, Dr. Kushiyaama, Dr. Cheng, Dr. Shinohara, Dr. Jaturapitpornchai, and Patcha for providing helpful comments on my research. I thank Ogushi-san for encouraging me and introducing SkymatiX, Inc. who provided me with a valuable opportunity to work in the industry. The discussions with Shinohara-san were always enjoyable and full of insights. Occasional meals with Tei (Cheng) are always refreshing. I also thank Manda-san who helps me with paperwork. I thank all the other members for providing a warm environment where I can always find security.

I am grateful to Dr. Kim, Dr. Liu, and Dr. Wang from AIST for giving me professional and technical suggestions on a weekly basis. I learned a lot from them about how to be a professional researcher. I also thank them for giving me a lot of suggestions related to my career and life. They have been indispensable to my research life.

I thank Takayama International Education Foundation for providing me with scholarships and many enjoyable moments. They undoubtedly have been the foundation of my life.

Finally, I want to thank my wife, Manna, who basically supports me in everything and balances my life. I thank my younger brother and parents who are always there whenever I need them. I thank my God for graciously giving me everything I have now and getting me through all the difficulties.

Contents

Abstract	iii
Synopsis	v
Acknowledgements	vii
1 Introduction	1
1.1 Background	1
1.2 Related Works	2
1.2.1 Collapsed building detection	4
Image-based methods	4
Point cloud-based methods	6
1.2.2 Deep learning for 3D point clouds	9
Deep Learning	9
Convolutional Neural Network	10
Multiview-based method for 3D point clouds	11
Voxel-based method for 3D point clouds	12
Point-based method for 3D point clouds	13
1.3 Objectives	21
1.4 Contributions and Dissertation Outline	24
2 Development of the Mashiki Dataset and Benchmarking	27
2.1 Introduction	27
2.2 Study Area and Data Source	28
2.2.1 Study area	28
2.2.2 Field survey	29
2.2.3 LiDAR point clouds	30
2.3 Dataset Development Workflow	31
2.3.1 Information extraction	31
2.3.2 GPS coordinate adjustment	35
2.3.3 Building annotation	35
2.3.4 Visual interpretation	36

2.3.5	Filtering	38
2.4	Statistics and Examples	38
2.5	Task Formulation	42
2.6	Benchmarking	43
2.6.1	Experiment settings	43
2.6.2	Evaluation metrics	44
2.6.3	Results	45
2.7	Summary	49
3	Development of Laplacian-Enhanced Neural Network (LE-Net)	51
3.1	Introduction	51
3.2	How to Represent Damage Numerically	53
3.2.1	Analysis of damage characteristics	53
3.2.2	Discrete Laplacian	55
3.2.3	Analysis of the discrete Laplacian concerning typical damage	55
3.3	Laplacian Unit	57
3.3.1	Disentangling convolution	59
3.3.2	Examples of w_{ij}	59
3.3.3	Formulation of Laplacian Unit	61
3.3.4	How complicated for a neural network to learn LU?	63
3.4	Laplacian-Enhanced Neural Network	64
3.4.1	LU-MLP and downsampling block	64
3.4.2	LU-upsampling block	67
3.4.3	Constructing LE-Net	68
3.5	Summary	68
4	Collapsed Building Detection Using LE-Net	69
4.1	Introduction	69
4.2	Performance and Design Analysis of LE-Net	70
4.2.1	Performance assessment	70
	Quantitative analysis	71
	Qualitative analysis	73
4.2.2	Analysis of LU	75
	Component analysis	75
	Effect of the selectivity function $\mathcal{F}$	77
4.2.3	Visual explanation	81
	SEG-GRAD-CAM	81
	Visualizing the decision-making of LE-Net	82

Failure modes	84
4.2.4 Computational efficiency	86
4.2.5 Robustness to the density variation	87
4.2.6 Effect of neighborhood size k	88
4.2.7 Damage mapping under a real-world scenario	88
4.2.8 Experiments and discussions on ground deformation in- duced building damage	93
4.3 Adaptability and Transferability of LE-Net	96
4.3.1 Haiti dataset	96
Study area and data sources	96
Dataset development workflow	98
Statistics and examples	104
Differences between the Mashiki and the Haiti dataset	106
4.3.2 Performance assessment	109
Evaluation framework and settings	109
Evaluation without adaptation	110
Evaluation of the adaptability	114
Evaluation of the transferability	116
4.3.3 Analysis and discussion on the performance gap between the Mashiki and Haiti results	118
4.4 Summary	121
5 Conclusions	125
5.1 Summary	125
5.2 Limitations	127
5.3 Future Work	128
A Additional Analyses	131
A.1 Extended Comparison and LE-Net Large	131
A.2 Comparison of Different Local Operators	132
A.3 Spectrum of building point clouds	133
A.4 Use of spatial geometry in the Laplacian Unit	135
B Application of LU to Other Tasks	137
B.1 Application to Shape Part Segmentation	137
B.2 Application to Scene Segmentation	137
B.3 Curvature Analysis	140

C	Using Risai Damage Grade as Ground Truth	143
C.1	Hand-Crafted Feature Calculation	143
C.2	Algorithms	144
C.3	Results and Discussion	145
C.3.1	“Background”, “Major and Story-collapsed” and “Other damage” classification	145
C.3.2	“Background”, “Story-collapsed” and “Other damage” classification	146
C.3.3	Separated damage classification	147
D	Publications Related to This Research	149
E	Detailed explanations of the discrete Laplacian	151
E.1	Finite difference approximation of 1D Laplacian	151
E.2	Image Laplacian to the point Laplacian	152
	Bibliography	155

List of Figures

1.1	The comparison between an aerial image and 3D point clouds of non-collapsed and collapsed buildings.	3
1.2	A general categorization of image-based CBD	7
1.3	Categorization of point cloud-based CBD methods.	9
1.4	A simple illustration of MLP in the context of point cloud analysis	13
1.5	Illustration of ReLU	14
1.6	Structure of PointNet	15
1.7	Detailed process of max-pooling	15
1.8	The architecture of PointNet++	17
1.9	Illustration of Group MLP module	18
1.10	Categorization of local aggregation operators	20
1.11	Dissertation outline	24
2.1	The study area for developing the Mashiki dataset	28
2.2	Example of the (simplified) recording sheet	30
2.3	Field survey results and extracted raw building GPS coordinates	31
2.4	Visualization of post-event LiDAR data used for developing the Mashiki dataset	32
2.5	The Mashiki dataset development workflow	33
2.6	GPS coordinates adjustment procedure	35
2.7	Annotation tool for developing the Mashiki dataset	37
2.8	Building annotation and visual interpretation procedure	38
2.9	Damage pattern chart	39
2.10	Distribution of collapsed and non-collapsed buildings in the Mashiki dataset	40
2.11	Examples of collapsed and non-collapsed buildings in the Mashiki dataset	41
2.12	Task formulation	43
2.13	Confusion matrix calculation	44
2.14	Illustrations of precision, recall, and IoU	45
2.15	Result of PointNet++ on the Mashiki dataset	46
2.16	Failure modes of PointNet++ on the Mashiki dataset	48

3.1	Comparison of non-collapsed and collapsed buildings	54
3.2	The illustration of the discrete Laplacian	56
3.3	The comparison of the distribution of discrete Laplacians (Δx_i in Equation 3.1) between collapsed and non-collapsed buildings	58
3.4	Visualizations of Laplacians from various filtering results	60
3.5	The computation of LU	61
3.6	The concept of the selectivity function $\mathcal{F}$	62
3.7	Detailed architecture of LE-Net	65
3.8	Network architecture of LE-Net	66
3.9	Illustration of the successive sampling process and the change of receptive fields.	67
4.1	The outline of Chapter 4	69
4.2	Qualitative results collapsed building detection on the Mashiki dataset	74
4.3	Effect of the selectivity function $\mathcal{F}$	78
4.4	The qualitative result of LE-Net without the selectivity function $\mathcal{F}$	79
4.5	The concept of SEG-GRAD-CAM	81
4.6	Visual explanation using SEG-GRAD-CAM	82
4.7	Failure modes of LE-Net on the Mashiki dataset	84
4.8	Robustness to the density variation	87
4.9	Effect of neighborhood size k	89
4.10	Illustration of seed point-based extraction	90
4.11	Overview of the Shimada area (Google Earth)	91
4.12	Result of CBD of Shimada area	92
4.13	Overview of the area with large ground deformations	94
4.14	Change of training and test region	95
4.15	Result of detection of a collapsed building in the presence of ground deformation	96
4.16	The study area of the Haiti dataset development	97
4.17	Classification of damage to buildings according to EMS-98	98
4.18	The LiDAR data used for developing the Haiti dataset	99
4.19	The overview of the Haiti dataset development workflow	101
4.20	Comparison among various damage scales	102
4.21	The effect of the majority filter	103
4.22	Visualization of buildings in the Haiti dataset	106
4.23	Roof types classification	107
4.24	Comparison of collapse patterns of wooden and concrete buildings	109

4.25	Confusion matrices of LE-Net without adaptation	110
4.26	The qualitative results of LE-Net without adaptation	112
4.27	The compilation of collapse forms	113
4.28	The failure modes of LE-Net without adaptation	113
4.29	The qualitative results of the adaptability experiments	116
4.30	The transferability of LE-Net with varying numbers of training samples	117
4.31	Performance difference between the Mashiki and Haiti results with varying numbers of training samples	119
4.32	Ambiguities in the definition of non-collapsed and collapsed build- ings in the Haiti dataset	120
4.33	Examples of ambiguities in the Haiti dataset	121
A.1	Result of Fourier analysis	134
B.1	Qualitative result of part segmentation	138
B.2	Qualitative result of scene segmentation	139
B.3	Visualization of curvatures using the ShapeNet Part dataset . . .	141
B.4	Visualization of curvatures using the S3DIS dataset	141
C.1	Classification results of “Background”, “Major and Story-collapsed” and “Other damage”	146
C.2	Classification results of “Background”, “Story-collapsed” and “Other damage”	147
C.3	Results of separated damage classification	148
E.1	Comparison of discrete image and point Laplacians	153

List of Tables

1.1	Detailed information of PointNet++ architecture	16
1.2	Strengths and weaknesses of point cloud-based CBD methods	22
2.1	Statistics of the Mashiki dataset	38
2.2	#points in training, validation, and test regions	40
2.3	The benchmarking result of the mainstream deep learning models (%)	45
3.1	Comparison of convolution, disentangled convolution, and Laplacian Unit	61
4.1	Comparison of overall performance (%) on the Mashiki dataset	71
4.2	Detailed comparison of LE-Net (Ours) and PointNet++	71
4.3	Percentage of “Good” detection of collapsed buildings	72
4.4	Result of design analysis	76
4.5	Comparison of computational efficiency	86
4.6	The number of buildings in each split in the Haiti dataset	105
4.7	The number of points in each split in the Haiti dataset	105
4.8	Visual roof types classification result using 50 random samples.	107
4.9	Structure type distribution of Mashiki Town	108
4.10	Structure type distribution of Port-au-Prince area, Haiti	108
4.11	The result without adapting to the Haiti dataset	110
4.12	The result without pretraining	115
4.13	The effect of pretraining with increasing numbers of training samples.	117
A.1	Extended comparison of overall performance.	131
A.2	Extended detailed comparison of LE-Net (Ours), LE-Net Large (Ours), and PointNet++	132
A.3	Comparison of different local operators.	133
A.4	Usefulness of 3D geometry (positions) in LU	135

List of Abbreviations

BoW	Visual- B ag-of- W ords
CBD	C ollapsed B uilding D etection
CNN	C onvolutional N eural N etworks
DNN	D eep N eural N etwork
FP	F eature P ropagation
FPS	F arthest P oint S ampling
kNN	k - N earest- N eighbor
LE-Net	L aplacian- E nhanced N eural N etwork
LiDAR	L ight D etection A nd R anging
LU	L aplacian U nit
MLP	M ulti L ayer P erceptron
RANSAC	R andom S ample C onsensus
UAV	U nmanned A erial V ehicle
RC	R einforced C oncrete
SA	S et A bstraction

Chapter 1

Introduction

1.1 Background

Earthquakes have been one of the most catastrophic disasters that affect mankind (Dong and Shan, 2013). One of the biggest threats that come from earthquakes is building damage. Building damage refers to various kinds of damage that possibly affects the well-being of people who live in them. In particular, detecting collapsed buildings are important because of their severe impact on the lives of residents and their financial situations. In fact, a majority of injuries and deaths result from building collapse in most earthquakes (Schweier and Markus, 2006). Therefore, detecting earthquake-induced collapsed buildings is crucial after an earthquake for humanitarian assistance and post-disaster recovery.

The most basic way to identify collapsed buildings is the field survey. During the field survey, trained investigators visit buildings directly to determine the damage level of the target buildings. However, conducting field surveys is difficult immediately after an earthquake because the locations of collapsed buildings are uncertain. In such situations, an approach that can rapidly detect collapsed buildings is necessary for an effective rescue. Fortunately, remote sensing techniques can provide a rapid and low-cost way to detect collapsed buildings on a large scale; therefore, it remains the most popular type of approach to collapsed building detection (CBD).

Among various types of data obtained through remote sensing techniques, the most frequently used one for CBD is the image. Remotely sensed images include images such as satellite images or aerial images which are captured by cameras mounted on an aircraft. The most traditional method for CBD using images is visual interpretation. Visual interpretation in general can obtain good results at the building level. However, it takes time and requires trained human operators, and thus it is unsuitable for rapid damage assessment. To reduce the workload, various automatic damage detection methods are proposed. Methods ranging from image enhancement to modern machine learning techniques have

been successfully applied to disaster images to achieve efficient CBD. Recently, motivated by the massive success in fields such as computer vision and natural language processing, deep learning techniques have been applied to CBD and have shown impressive performance in detecting collapsed buildings using images.

On the other hand, point clouds obtained by airborne laser scanning are especially fitting to CBD because of their ability to record building heights with high accuracy. Unlike methods like Structure-from-Motion and oblique images which requires massive numbers of images or expensive computation to generate point clouds from images (Liu et al., 2022), laser scanning can rapidly acquire point clouds at a large scale. Compared with images that can only show the top-down 2D nature of buildings, the additional height information enables the analysis of the 3D nature of buildings. This is vital because some collapsed forms are extremely difficult to identify in 2D space but can be identified relatively easily in 3D space. Such an example is shown in Figure 1.1. Roughly speaking, the height information is useful in three ways: 1) the building height itself is a useful indicator for detecting collapse. As can be seen from the figure, the non-collapsed building (the top row) is usually taller than the collapsed one (the middle row); 2) the height information can also be used to detect collapsed buildings that cannot be detected easily by images (the middle row); 3) As shown in the bottom row, the height information also helps to identify building with significant structural failure but withstand total collapse. Although there exist some approaches that detect collapsed buildings using point cloud, including pre-post event change detection (Moya et al., 2018) and geometric analysis of post-event point clouds (Rehor et al., 2008; He et al., 2016b), the approach that adopts deep learning techniques are lacking. Given the advantages of 3D point clouds and the state-of-the-art performance of deep learning techniques, we believe that the combination of these two elements can significantly advance automatic CBD and provide valuable insights for future research.

1.2 Related Works

In this section, we review the related works to provide the basic technical background for this research so that the contributions made by this study can be clearly identified. The section begins with a review of existing CBD methods. Specifically, image-based methods are introduced because they are the most common approach for CBD. Then, point cloud-based methods are introduced.

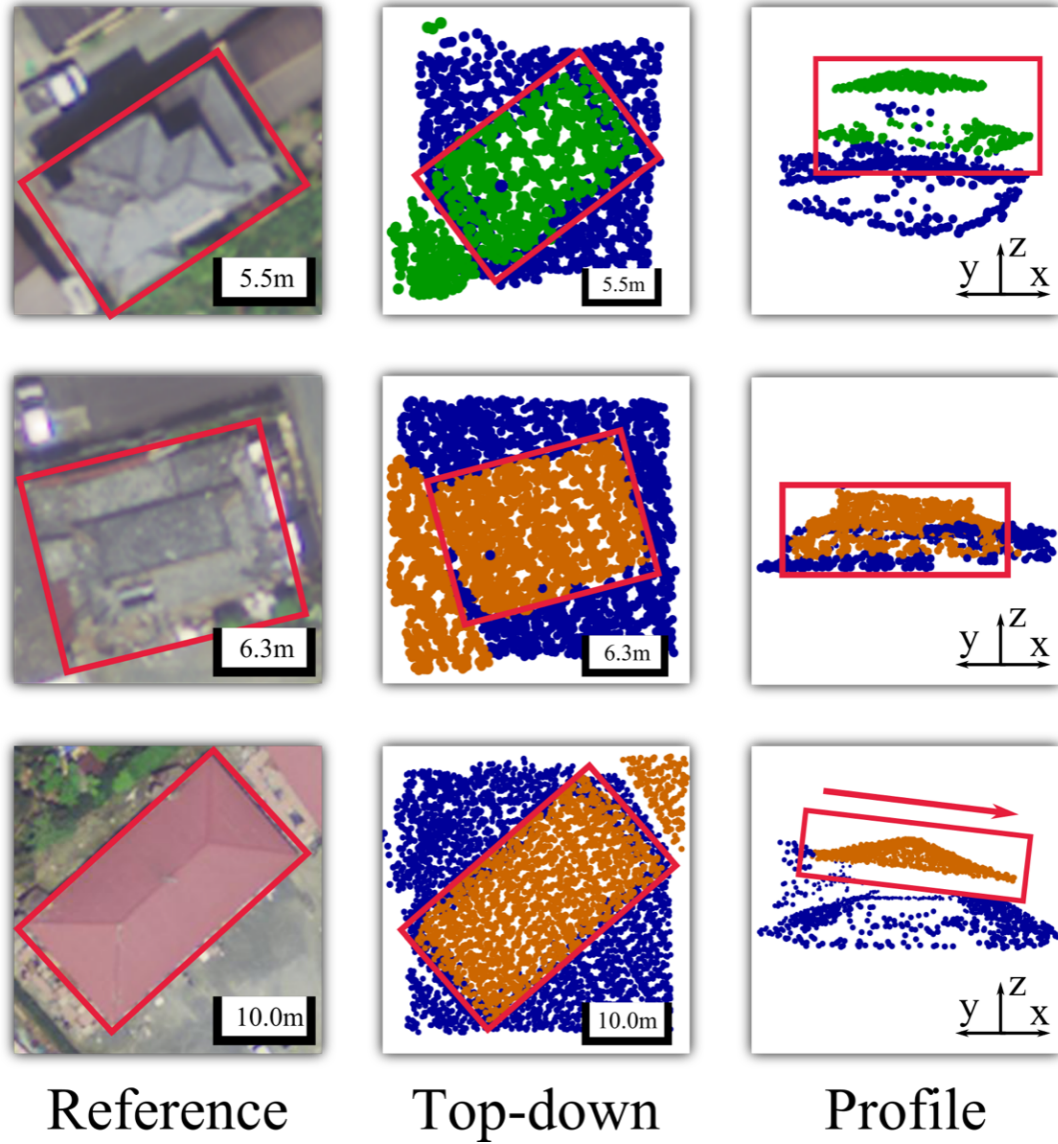


FIGURE 1.1: The comparison between an aerial image and 3D point clouds of non-collapsed and collapsed buildings. Top row: a non-collapsed building. The second and third rows: collapsed buildings. The “Reference” column shows the aerial images (provided by Cheng, Satoh, and Matsuoka (2018)) of the buildings for reference purposes. The “Top-down” column shows the top-down view of the point clouds while the “profile” column shows the profile view of point clouds. Green, brown, and blue points represent points of the non-collapsed building, collapsed building, and background, respectively. The red boxes indicate the target building and the red arrow indicates the inclination. As can be observed, detecting collapsed buildings is challenging from the aerial images due to the lack of height information.

Next, we provide the technical background of deep learning-based point cloud processing techniques.

1.2.1 Collapsed building detection

Image-based methods

Satellite and aerial images are the most frequently used types of data thanks to their large coverage and relatively low cost (Tong et al., 2012). The earliest approach for damage detection is visual interpretation. Gamba and Casciati (1998) improves the detection accuracy with the help of the geographic information system. Saito et al. (2004) extracts collapsed buildings using three satellite images that are taken before and after the event. Yamazaki, Yano, and Matsuoka (2005) identifies destroyed buildings with a single QuickBird panchromatic image. Visual interpretation is accurate because human operators interpret and determine the damage level of the building directly. However, it is time-consuming due to the large number of buildings to be examined. Furthermore, it is challenging even for human operators to identify the types of collapse that cannot be easily identified from 2D images (see Figure 1.1).

To achieve efficient detection, automatic image processing techniques are introduced to CBD extensively. For instance, numerous methods are designed to handle co-registered pre-event and post-event images. Kohiyama and Yamazaki (2005) detect the damaged regions using Terra-ASTER images that have 15m resolution. Their method models the fluctuation of the digital numbers as random variables based on pre-event images. The deviation of digital numbers in the post-event image is calculated and converted into confidence levels, showing the surface changes due to the earthquake. However, it is sometimes difficult to determine whether the detected damage is caused by collapsed buildings or dusty roads due to the coarse resolution. Sakamoto et al. (2004) proposes to detect geographical changes by non-linear mapping using QuickBird images. In this method, the post-event data is nonlinearly mapped onto the pre-event image and the resulting matching scores are used to estimate the damage grade. Moreover, Miura, Modorikawa, and Chen (2011) uses the texture characteristics to detect damaged buildings using the pre-event QuickBird and post-event WorldView-2 images. Synthetic Aperture Radar (SAR) is also frequently used because of its high sensitivity to height changes. Matsuoka and Yamazaki (2004) find that the damaged areas show significantly lower backscattering coefficients and intensity correlations calculated from pre- and post-event SAR images. An automatic discriminant analysis method is developed subsequently, and it is

further revised and applied to different earthquakes (Matsuoka and Yamazaki, 2005).

Since pre-event data are not always available, damage detection using only post-event data is generally preferred. For example, some methods use edge and texture information to characterize and detect collapsed buildings. Sugiyama and Abe (2002) detect the damaged areas based on the assumption that damaged areas contain more edges whose orientations follow a certain distribution. Vu, Matsuoka, and Yamazaki (2005) developed a method that can detect damaged buildings using post-event IKONOS and QuickBird images with an accuracy of 80%. Yamazaki, Vu, and Matsuoka (2007) combine the edge-based textures, multispectral gray tone, and spatial relationship from morphological scale-space and infer the damage level by measuring the degree of overlap between detected debris and retained building areas.

Motivated by the tremendous amount of available data, machine learning techniques are successfully applied to CBD. For instance, Gerke and Kerle (2011) use the conditional random field and AdaBoost for damage classification. In their approach, airborne oblique, multi-perspective Pictometry data are used and the entire procedure is split into two steps: 1) supervised classification into facades, intact roofs, destroyed roofs, and vegetation with 22 image features, and 2) combining the results from different views into building-wise damage scores. Cooner, Shao, and Campbell (2016) evaluate the effectiveness of multilayer feedforward neural networks, radial basis neural networks, and Random Forests in detecting damaged buildings. In their work, various textural and structural features are investigated as key variables. They show that the multilayer feedforward network is able to achieve an error rate lower than 40% in detecting collapsed buildings and spatial features are far more important in terms of classification than spectral information. Vetrivel et al. (2016) develops a method that performs grid-level damage classification based on Visual-Bag-of-Words (BoW) and it is compared to several conventional supervised classifiers with commonly used features.

Recently, the massive success of deep learning and convolutional neural networks (CNN) (Krizhevsky, Sutskever, and Hinton, 2012) has triggered its application to CBD. Deep learning techniques are generally preferred over conventional machine learning methods like Support Vector Machines and Random Forests because they do not need hand-crafted features for input and perform generally better than them. Its applicability is quickly proven in remotely sensed image processing (Castelluccio et al., 2015; Maggiori et al., 2016), which guaranteed its success in detecting collapsed buildings as well. For instance, Bai et al.

(2017) propose a deep learning-based framework damage recognition method using post-event SAR images. Their method has achieved an accuracy of about 80% and its applicability in time-critical situations is validated. In the work of Kamagatani and Matsuoka (2017), a CNN is constructed to identify damaged buildings using aerial photographs. In addition, estimated seismic intensities at the location of buildings are combined with the input of CNN to enable a more comprehensive understanding of damage. Their result suggests that a detailed estimation of building damage based on damage probability is possible. Ci, Liu, and Wang (2019) proposes a method that is a combination of a CNN and ordinal regression, which is motivated by the ordinal nature of damage grades. Hamaguchi, Sakurada, and Nakamura (2019) adopts a method that learns the disentangled feature representation so that rare events can be detected and its effectiveness is validated comprehensively on four change detection datasets. Naito et al. (2020) construct a model based on bag-of-visual-words and CNN and compared them. Notably, the models classify building damage into four damage categories. Their result shows that CNN is a better classifier that can achieve accuracy ranging from 66% to 81%. The introduction of xBD dataset (Gupta et al., 2019), a large-scale dataset containing various disaster-affected buildings from different satellites, has further boosted the trend. Weber and Kané, 2020 analyze the problem framing, data processing, and training procedures which are specifically helpful for CBD using the xBD dataset. As a result, their insights lead to substantial improvement over the xBD baseline models. Gupta and Shah (2020) constructs a CNN with a new loss function that can simultaneously segment buildings and perform building-level damage assessment in an end-to-end fashion. On the other hand, Shen et al. (2021) propose a two-stage method in which buildings are localized in the first stage and classified in the second one. Moreover, they propose a cross-directional attention module to specifically exploit the correlation between pre- and post-event images.

A general categorization of image-based methods is shown in Figure 1.2.

Point cloud-based methods

Airborne Light Detection And Ranging (LiDAR) systems can rapidly capture information about buildings in the form of 3D point clouds. Captured point clouds possess precise height information of buildings; therefore, they are especially useful for detecting collapsed buildings. Taking advantage of this, varieties of methods are proposed to detect collapsed buildings through geometric analysis. For example, Rehor and Bähr (2006) perform the segmentation of planar surfaces and classify Delaunay-based triangulated irregular network (TIN)

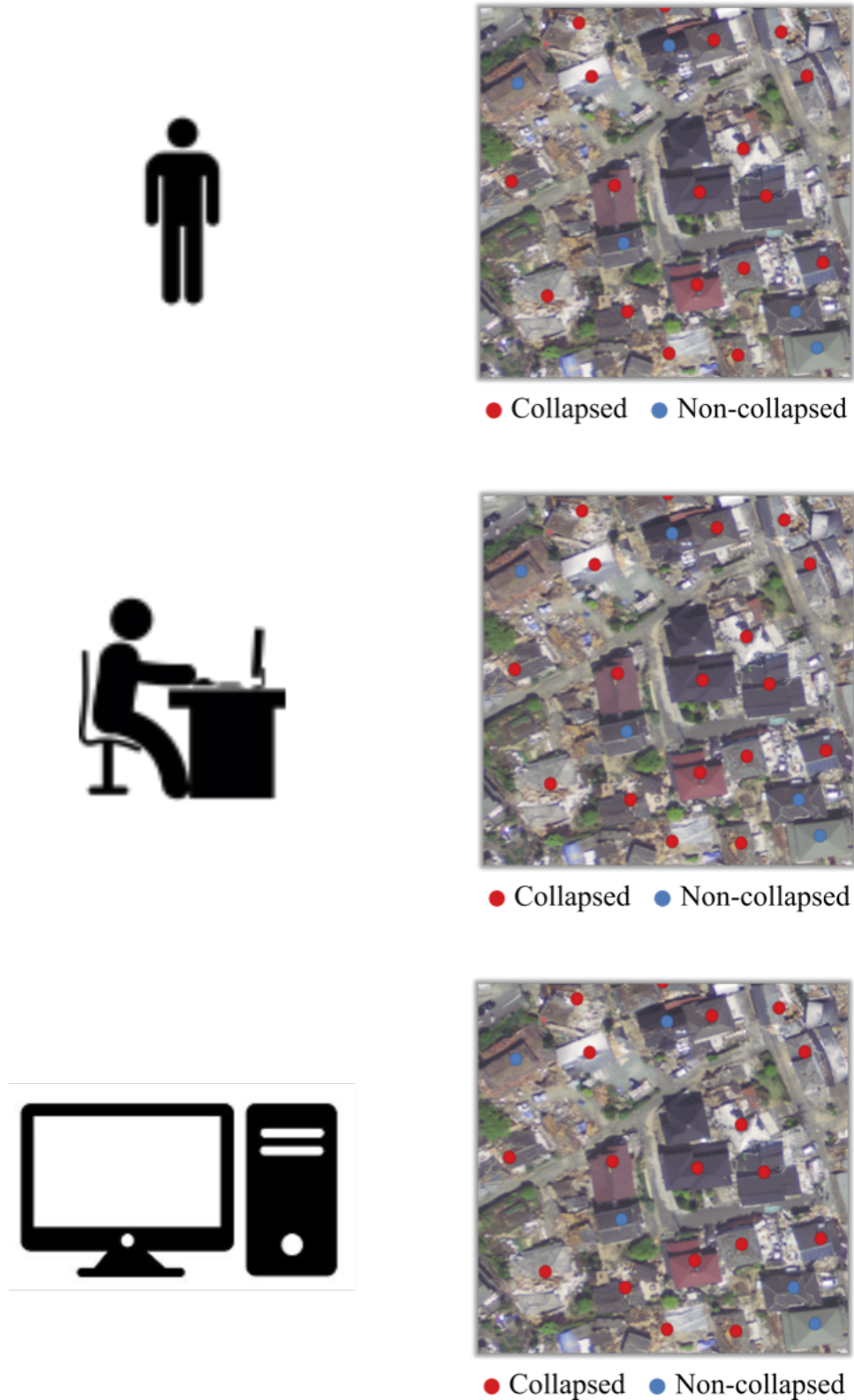


FIGURE 1.2: A general categorization of image-based CBD. Top: visual interpretation of collapsed buildings by human operators. Middle: CBD using image processing techniques such as texture analysis and conventional machine learning using hand-crafted features. Bottom: CBD using deep learning without hand-crafted features.

segments into undamaged and damaged elements. In their method, the extracted debris is treated as damaged parts and shows that clear assumptions can be made for damaged buildings. In Rehor et al. (2008), to detect building damage, the random sample consensus (RANSAC) and region-growing method are applied to digital surface models. As a result, they show that the region-growing algorithm is more fitting to damage detection compared to RANSAC. Labiak et al. (2011) develops a slope threshold method for detecting damaged points where they assume that points in non-damaged roofs have constant slopes whereas damaged roof surfaces present varying slopes. Yonglin, Lixin, and Zhi (2010) compute the geometric axis of a flat or symmetric roof and detect inclined roofs by calculating angles between the geometric axis and vertical line using empirical thresholds. Fernandez Galarreta, Kerle, and Gerke (2015) propose an object-based image analysis for damage assessment. The detailed point clouds are generated using the unmanned aerial vehicle (UAV) multiview images and features derived from local covariance analysis are used to assess the damage. He et al. (2016b) propose a contour-cluster-based 3D shape descriptor that can detect both surface roof and structural damage from airborne 3D point clouds. Their method is based on the assumption that intact roofs show compact and regular contour patterns whereas damaged roofs show jagged and irregular contours.

Change detection can also be applied to point cloud-based CBD. Moya et al. (2018) perform change detection using a pair of airborne LiDAR-derived DSMs. The obtained distribution of the collapsed buildings in the study area shows good correspondence with the field survey conducted after the event. Their findings also show that the change in average elevation within building footprints is the most important factor for detecting collapsed buildings. However, the change detection using LiDAR for CBD is difficult to apply in practice because of the scarcity of pre-event LiDAR data.

Machine learning approaches are also proven useful in detecting building damage using point clouds. In the work of Elberink et al. (2011), the segmentation is performed to group points into planar segments. For each segment, various geometric and radiometric features such as planarity are computed. Using those attributes, the final classification is performed using maximum entropy modeling. Khoshelham and Oude Elberink (2012) also presents a segment-based approach to detecting damaged building roofs in airborne point clouds. The methods first perform segmentation and group points into planar segments. Then, the feature extraction is followed by supervised classifications using classifiers such as the linear discriminant classifier. Khoshelham, Oude Elberink,

and Xu (2013) also performs segment-based methods and reveals that feature selection improves classification accuracy when the number of training samples is few. Vetrivel et al. (2015) detects damaged regions by detecting and classifying gaps that are present in the airborne oblique images generated by point clouds. They assume that the gaps are related to physical damage and image artifacts and understanding them can be helpful to identify damaged regions. Their method successfully delineates buildings from a point cloud and classifies the present gaps. Subsequently, the features are extracted and damage mapping is performed using Support Vector Machines and Random Forests, resulting in 95% correctly identified damage regions.

A simple categorization of the point cloud-based CBD methods is shown in Figure 1.3.

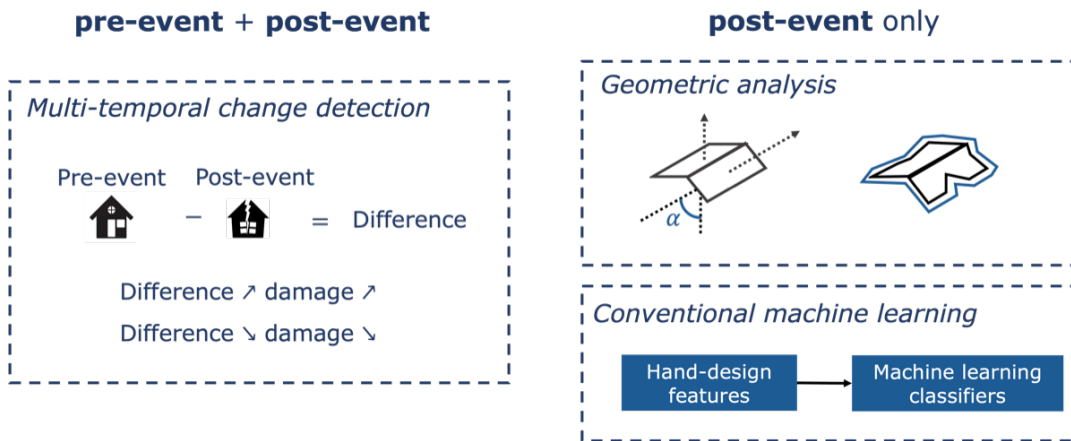


FIGURE 1.3: Categorization of point cloud-based CBD methods.

1.2.2 Deep learning for 3D point clouds

Deep Learning

Deep learning, which is a subset of machine learning, is inspired by how information is processed in the human brain. Therefore, deep learning algorithms are commonly referred to as deep neural networks (DNNs). DNNs consist of multiple layers and each layer comprises neurons, which are the basic computation unit of neural networks with learnable parameters. The input data is passed to the neural networks and go through all layers. Whenever data goes through a layer, it is transformed by a layer, obtaining a different representation of the input data (Alzubaidi et al., 2021). The last layer of DNNs is usually a layer that performs a specific task. For example, if the task is to classify

an input image, the last layer would be a classification layer that produces the scores on which the decision can be based. Given input data and output ground truth, DNNs are optimized by training. This training process is usually done using backpropagation that propagates the error back to each computation unit of DNNs (Rumelhart, Hinton, and Williams, 1986). The initial error signal is generated by calculating the deviation between the output of DNNs and ground truth, and it is subsequently propagated to each neuron of DNNs using the chain rule. At each neuron, their parameters are adjusted by gradient descent so that the update of parameters can make the error smaller. By repeating this process numerous times, DNNs are “trained” to perform a task. One particularly nice characteristic of DNNs is that they do not need to transform input data into features in a hand-crafted manner before feeding data into the networks. A good feature extractor requires careful engineering and considerable domain expertise (LeCun, Bengio, and Hinton, 2015). For example, the height from the ground may be a useful feature that can be used for detecting buildings; therefore, “the height from the ground” feature must be computed by human operators before feeding it to algorithms. Conversely, DNNs are able to process the raw data and learn the suitable representations of data automatically through training. This trait enables an end-to-end optimization process where only the least amount of human intervention is needed. This process, of course, needs a tremendous amount of data from which DNNs can learn useful patterns and huge computation power. These conditions are met by the introduction of big datasets such as ImageNet (Deng et al., 2009) and advanced computation hardwares (Krizhevsky, Sutskever, and Hinton, 2012). The impact of deep learning has reached many scientific fields including but not limited to image recognition (Krizhevsky, Sutskever, and Hinton, 2012; Farabet et al., 2012), speech recognition (Hinton et al., 2012a), natural disaster forecasting (Nevo et al., 2019), and drug discovery (Chen et al., 2018). In particular, it is reported that the performance of DNNs has beaten that of humans on tasks such as image classification. Based on those facts and the unprecedented progression of deep learning, it is estimated that DNNs are going to be able to do everything eventually.

Convolutional Neural Network

CNN is a particular branch of DNNs that mainly deals with image recognition. It has become famous in the context of computer vision since its huge success in the ImageNet Large-Scale Visual Recognition Challenge in 2012 (Krizhevsky, Sutskever, and Hinton, 2012). The work of Krizhevsky, Sutskever, and Hinton

(2012) is a breakthrough that used the CNN to almost halve the error rate for object recognition, which triggered a rapid adoption of deep learning by the computer vision community (LeCun, Bengio, and Hinton, 2015). CNN's success mainly comes from two sources: 1) advanced computation hardware and software that support massive computation and learning capacity and 2) assumptions on the nature of data, i.e., images, which are stationarity of statistics and locality of pixel dependencies (Krizhevsky, Sutskever, and Hinton, 2012). CNN also adopts equivalent representations, sparse interactions, and parameter sharing (Goodfellow, Bengio, and Courville, 2016), making CNN easier to train and robust to overfitting. A typical CNN consists of several types of layers/functions: 1) convolution layers that are responsible for feature learning, 2) pooling layers that are responsible for feature aggregation, 3) activation functions that are responsible for adding nonlinearity to features, and 4) a classification/regression layer which performs the task. These types of layers are flexibly stacked and combined to construct various types of CNNs. Although there might be infinite combinations of those layers, there are several architectures that are widely acknowledged by researchers. They are but are not limited to 1) AlexNet (Krizhevsky, Sutskever, and Hinton, 2012) which first adopts Dropout (Hinton et al., 2012b) and ReLU (Nair and Hinton, 2010); 2) VGG (Simonyan and Zisserman, 2014) which improves the performance by substantially increasing the depth of the network and adopts the small filter sizes; 3) GoogLeNet (Szegedy et al., 2015) that introduces the combination of different filter sizes; 4) Inception-V4 (Szegedy et al., 2017) that introduces divided transforms and integration concepts; 5) ResNet (He et al., 2016a) that constructs extremely deep architectures while preventing overfitting by novel residual connections; and 6) SE-Net (Hu, Shen, and Sun, 2018) that introduces the channel attention mechanism which is used to rescale the features. Those architectures are not only influential in their own fields but are also transferred and adapted to other fields like 3D point cloud processing.

Multiview-based method for 3D point clouds

CNNs are extremely effective in dealing with structured data such as images; however, they cannot be directly applied to 3D point clouds due to their unstructured nature. In order to cope with the problem, Multiview-based methods project a 3D shape into multiple views and extract view-wise features followed by aggregation to perform feature learning. Although multiview processing can be done by CNNs, how to aggregate those views into a global representation becomes a key challenge for this type of method. Su et al. (2015) aggregates

the multiview features by the max-pooling. However, it is argued that the max-pooling only extracts the maximum feature, thus leading to information loss. Yu, Meng, and Yuan (2018) uses a harmonized bilinear pooling to produce a compact global descriptor. Yang and Wang (2019) propose a method that effectively models the interconnecting of different viewpoints, resulting in reinforced individual view information. Wei, Yu, and Sun (2020) considers the multiple views as different graph nodes. Graph convolutions, non-local message passing, and selective view sampling are utilized for feature learning on constructed graphs features from all levels are aggregated by max-poolings into a global feature vector. Recently, Hamdi, Giancola, and Ghanem (2021) tackles the problem of heuristically set viewpoints and proposes to learn those viewpoints.

Despite that these methods can inherit 2D CNNs from image processing directly, multiview-based methods lose information by projection.

Voxel-based method for 3D point clouds

On the other hand, 3D point clouds can also be converted to 3D grids, namely 3D voxels. The conversion makes it possible to extend 2D CNNs to 3D CNNs. For instance, Maturana and Scherer (2015) introduce volumetric Occupancy Grid representation and combines it with 3D CNNs. As a result, efficient point cloud labeling is achieved. Wu et al. (2015) propose a convolutional deep belief-based 3D ShapeNets to learn various 3D shapes. The method learns to model those shapes as a probability distribution of binary variables on voxel grids. Although the above methods can use matured CNN architectures, they typically incur significant computation costs because their memory footprint increases cubically with the resolution, making them less scalable. To cope with this problem, Riegler, Osman Ulusoy, and Geiger (2017) partitions a point cloud using an octree structure hierarchically so that a scene can be represented with several octrees along a grid. Wang et al. (2017) propose O-CNN, an Octree-based CNN for 3D shape classification. The network receives the average normal vectors computed in the finest octants as input and 3D CNN is applied to octants occupied by the shape. Consequently, OctNet requires much less memory and running time compared to a baseline with dense input grids as input. Le and Duan (2018) propose a method that combines both the point and grid representations for efficient processing. Within each embedding volumetric grid, a fixed number of points is sampled. Therefore, geometric details of points can be extracted by applying 3D convolutions to them.

Point-based method for 3D point clouds

Different from multiview-based methods or voxel-based methods, point-based methods can handle raw point clouds without any projection. This is desirable because it can avoid the loss of fine-grained geometry due to projections. However, the unstructured nature of point clouds must be dealt with in order to consume raw point clouds. The pioneering work is PointNet (Qi et al., 2017a). The key innovations of PointNet are the use of per-point shared multilayer perceptrons (MLPs) and the symmetric function (e.g., max-pooling). The above two functions naturally solved the aforementioned problems because they are both set operations and permutation-invariant, meaning that the output does not change when the input point order is changed. The symmetric function, on the other hand, is also invariant to the point order, and thus it is suitable to point cloud processing.

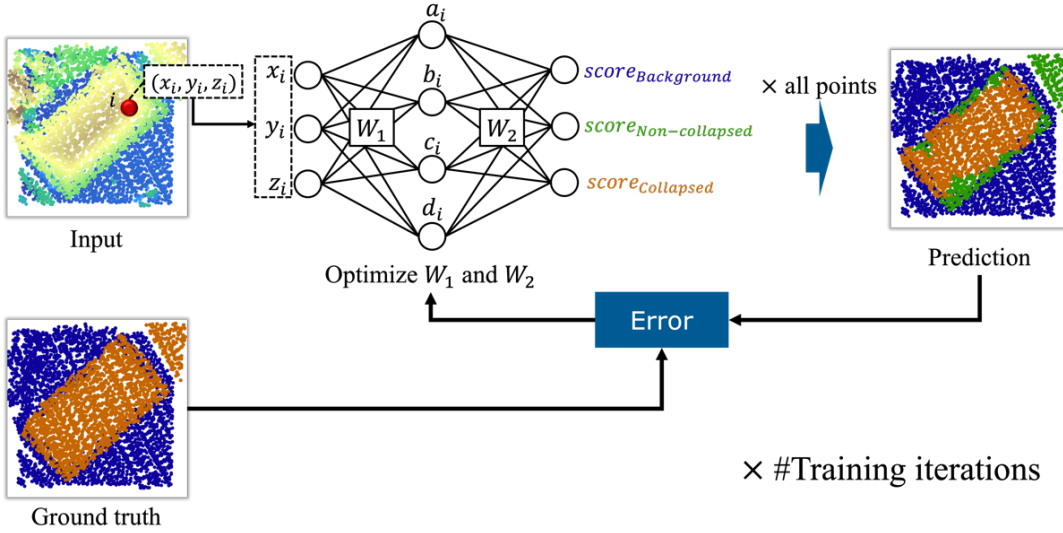


FIGURE 1.4: A simple illustration of MLP in the context of point cloud analysis. An MLP transforms the input feature of a point to class scores with multiple layers of nonlinear transformations.

Here we first provide an intuitive example to explain how MLP, the core of PointNet, works in detail. As shown in Figure 1.4, an MLP is defined as a fully-connected neural network that receives an input feature of a point i and produces an output feature for this point. An MLP has at least three layers including input, intermediate, and output layers. Using the example from the above figure, the input feature is $f_i^{in} = (x_i, y_i, z_i)$, the intermediate feature is $f_i^{mid} = (a_i, b_i, c_i, d_i)$, and the output one is $f_i^{out} = (score_{Background}, score_{Non-collapsed}, score_{Collapsed})$, which are scores for each class. In this context, a 3-layer MLP can be expressed

mathematically as follows:

$$f_i^{out} = \sigma(W_2 \cdot f_i^{mid}) \quad \text{where} \quad f_i^{mid} = \sigma(W_1 \cdot f_i^{in}). \quad (1.1)$$

where $W_1 \in \mathbb{R}^{4 \times 3}$ and $W_2 \in \mathbb{R}^{3 \times 4}$ are trainable linear transformations and $\sigma(\cdot)$ is a nonlinear activation function. Though many activation functions exist, we use Rectified Linear Unit (ReLU) ($\sigma = \max\{x, 0\}$) for our default activation function in this study. ReLU is shown in Figure 1.5. Moreover, here we omit

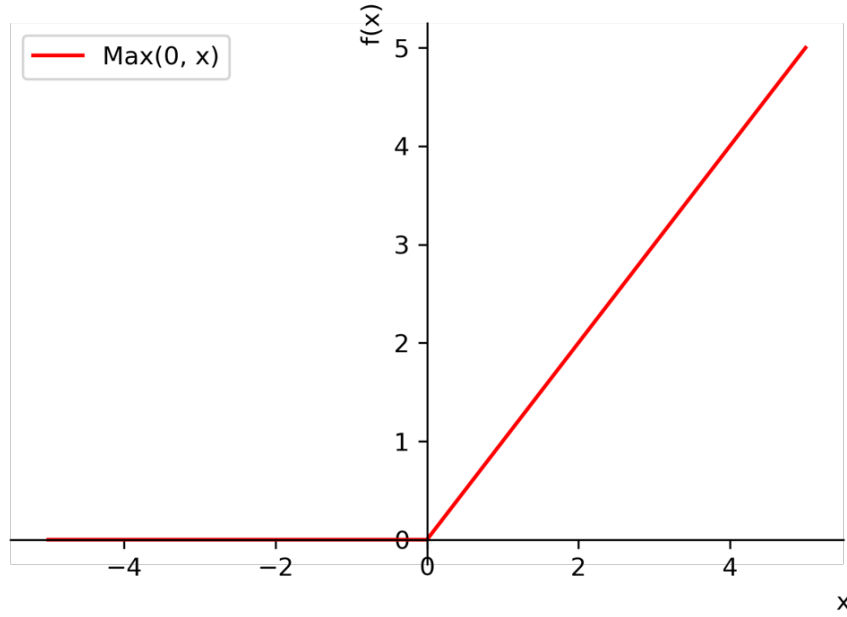


FIGURE 1.5: Illustration of ReLU.

biases and common functions like Batch Normalization (Ioffe and Szegedy, 2015) for simplicity.

We describe the internal structure of PointNet in Figure 1.6. As shown in the figure, PointNet repeatedly applies (shared) MLPs to input points features. Then, the global feature (z_{pool}) of the input point cloud is obtained by applying the max-pooling, which is a symmetric function. Specifically, max-pooling is a function used for information aggregation. The detailed procedure of max-pooling is shown in Figure 1.7. The resulting global vector contains per-channel maximum aggregated from all points. Therefore, z_{pool} describes the global characteristics of the input point cloud and thus it can be used to perform suitable tasks such as classification.

Despite its simplicity, PointNet performs exceedingly well in various point cloud-related tasks.

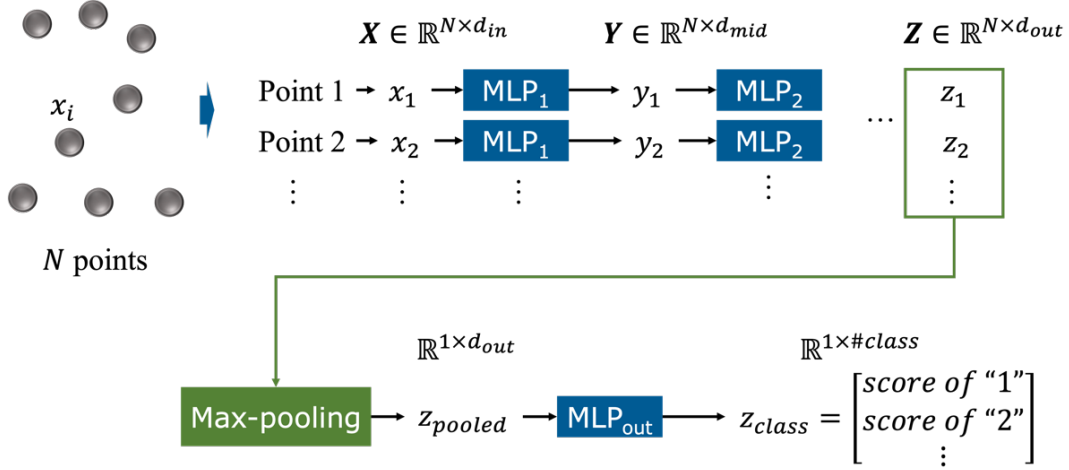


FIGURE 1.6: Structure of PointNet (Qi et al., 2017a). PointNet applies MLPs to input point features individually and aggregates information from all points using max-pooling.

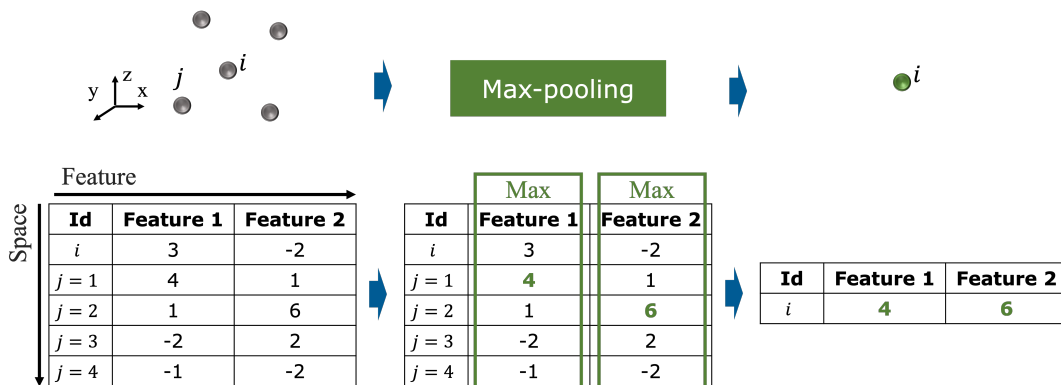


FIGURE 1.7: Detailed process of max-pooling. Max-pooling applies the max function to each channel. The resulting vector holds the per-channel maximum.

PointNet++ (Qi et al., 2017b) is a significant extension of PointNet. PointNet by nature does not capture the local structure of point clouds. However, as proven by the tremendous success of CNN, exploiting local structures/patterns are important for improving performance and generalizability. Therefore, PointNet++ introduces a hierarchical framework in which local patterns are kept abstracted along the hierarchy. In the following, we introduce the segmentation version of the architecture of PointNet++ in detail as this work uses PointNet++ extensively. The overall architecture of PointNet++ is shown in Figure 1.8. PointNet++ adopts the typical encoder-decoder architecture like

Layer	#point	Feature dim.	Operation
Encoder 1	N	$\mathbb{R}^{N \times 3} \rightarrow \mathbb{R}^{N \times d}$	Group MLP
Encoder 2	$\frac{N}{2}$	$\mathbb{R}^{\frac{N}{2} \times d} \rightarrow \mathbb{R}^{\frac{N}{2} \times 2d}$	Down-sampling, Group MLP
Encoder 3	$\frac{N}{4}$	$\mathbb{R}^{\frac{N}{4} \times 2d} \rightarrow \mathbb{R}^{\frac{N}{4} \times 4d}$	Down-sampling, Group MLP
Encoder 4	$\frac{N}{8}$	$\mathbb{R}^{\frac{N}{8} \times 4d} \rightarrow \mathbb{R}^{\frac{N}{8} \times 8d}$	Down-sampling, Group MLP
Decoder 1	$\frac{N}{4}$	$\mathbb{R}^{\frac{N}{4} \times 2d} \rightarrow \mathbb{R}^{\frac{N}{4} \times 4d}$	Up-sampling, MLP
Decoder 2	$\frac{N}{2}$	$\mathbb{R}^{\frac{N}{2} \times d} \rightarrow \mathbb{R}^{\frac{N}{2} \times 2d}$	Up-sampling, MLP
Decoder 3	N	$\mathbb{R}^{N \times 2d} \rightarrow \mathbb{R}^{N \times d}$	Up-sampling, MLP
Classification	N	$\mathbb{R}^{N \times d} \rightarrow \mathbb{R}^{N \times C}$	MLP

TABLE 1.1: Detailed information of PointNet++ architecture.
 C denotes the number of classes.

U-Net (Ronneberger, Fischer, and Brox, 2015). It first downsamples the input point clouds to perform multi-scale learning. The major component that is responsible for feature learning is the Group MLP module. The idea of the Group MLP module is illustrated in Figure 1.9. Specifically, for a center point, the Group MLP module uses the k-nearest-neighbor (kNN) algorithm to collect its neighbors. An MLP is subsequently applied to each point of the neighborhood and aggregated by max-pooling. Formally, let $p_i \in \mathbb{R}^3$, $f_i \in \mathbb{R}^d$ denote the 3D coordinates and the feature of point i . Further, let $\mathcal{N}(i)$ denote the neighborhood gathered by kNN. The Group MLP can be expressed as:

$$f_i^{out} = \max_j (\{MLP(p_j - p_i \parallel f_i) \mid j \in \mathcal{N}(i)\}), \quad (1.2)$$

where $f_i^{out} \in \mathbb{R}^{d_{out}}$ denotes the output feature, $\max$ denotes the max-pooling, and $\parallel$ denotes the concatenation operation. $MLP : \mathbb{R}^{3+d} \rightarrow \mathbb{R}^{d_{out}}$ is an MLP. The Group MLP module uses the relative position $p_j - p_i$ in combination with the feature to encode both geometry and semantics. Then, the output feature is halved (down-sampled) to obtain a sparser point cloud for the next layer. By doing this, the next kNN can gather farther neighbors with less computation,

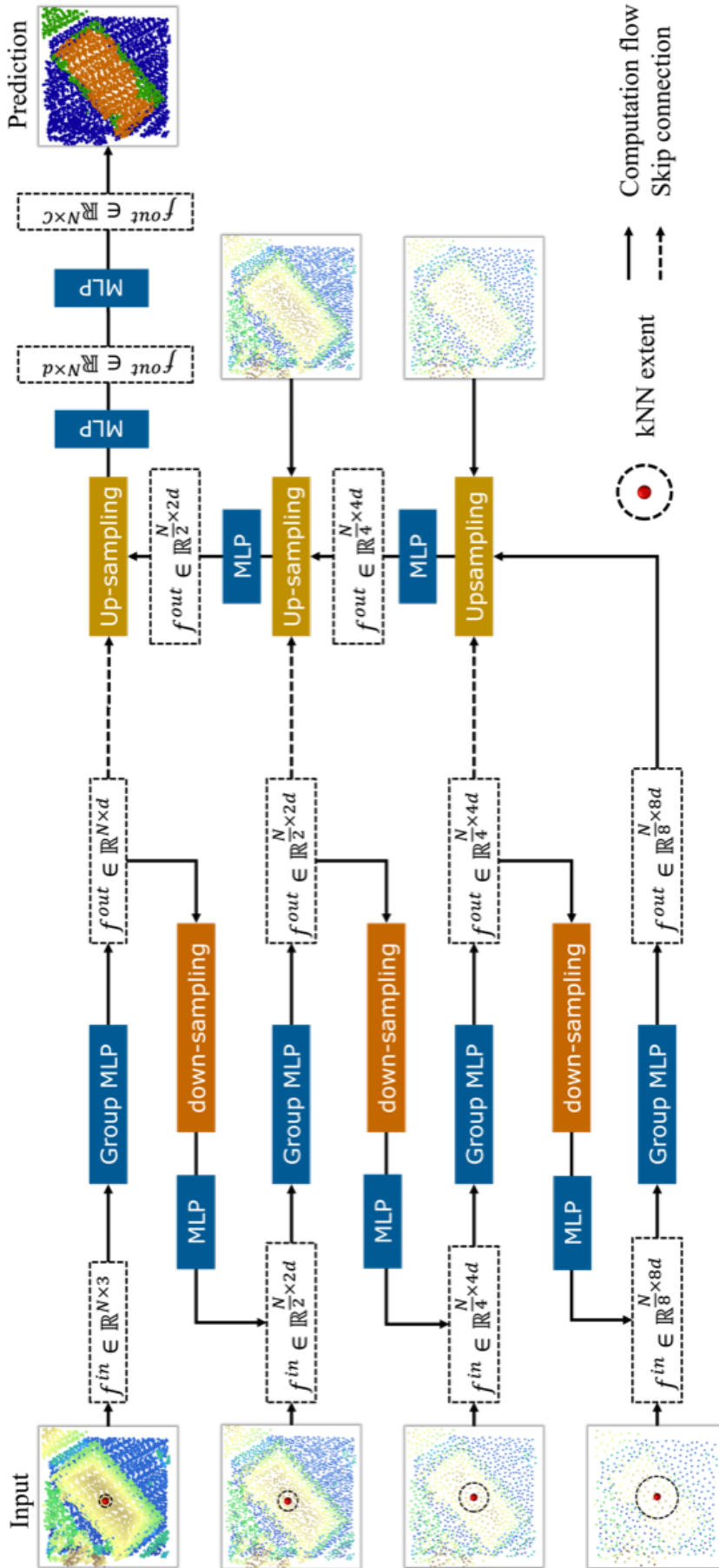


FIGURE 1.8: The architecture of PointNet++. N denotes the number of input points. d denotes the dimension of the input feature.

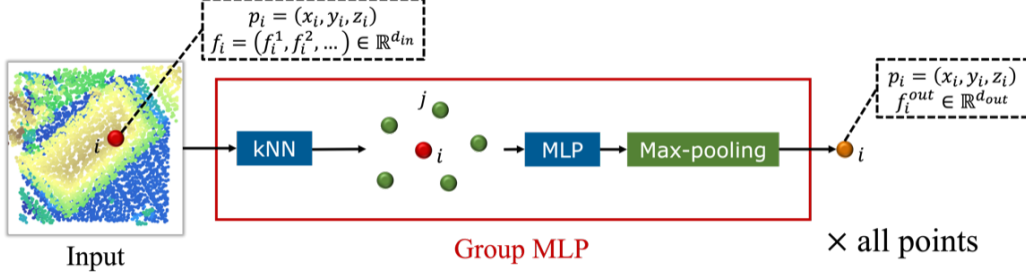


FIGURE 1.9: Illustration of Group MLP module.

achieving efficient hierarchical learning. The dimension of the feature is doubled after a Group MLP module so that the features can record increasingly more information while reducing the number of points. The output of the encoder is the sparse point clouds with high-dimensional features, which capture the global as well as local characteristics of point clouds.

As we need a full-resolution point cloud with per-point class categories as output, the features need to be reconstructed for those points missed due to the down-sampling process. PointNet++ adopts a similar strategy as U-Net which reconstructs point features by interpolation assisted by skip connections from the encoder. Concretely, a sparser level point cloud feature is interpolated to the denser one by inverse distance-weighted interpolation. Subsequently, the feature from the same level of the encoder is concatenated to the interpolated features and transformed by an MLP. Formally, let f_i^{dense} denote the feature to be reconstructed. Further, let f_i^{sparse} denote the input sparse point feature and $f_i^{dense_{enc}}$ the The reconstruction process can be described as follows:

$$f_i^{dense} = MLP \left(Interp(f_i^{sparse}) \parallel f_i^{dense_{enc}} \right). \quad (1.3)$$

By repeating this procedure, PointNet++ recovers the full-resolution point clouds with their features, enabling per-point predictions.

PointNet++ has become the standard network architecture for 3D point cloud processing. Since then, the development of local aggregation operators, i.e., local learners, become a hot topic in point cloud processing. Based on PointNet++, several extensions are proposed. They are typically categorized as Pointwise-MLP methods. For example, Zhang, Hua, and Yeung (2019) uses concentric shells to enhance the local feature learner. It partitions the local group into different shells and applies independent shared MLPs and symmetric functions. Lan et al. (2019) projects local points onto canonical basis vectors and performs independent transformations on those vectors. Qiu, Anwar, and Barnes (2021) utilizes error-correcting feedback to capture the local feature more

comprehensively. Qian et al. (2022) enhances PointNet++ directly by exploiting advanced training strategies and architecture engineering. DGCNN (Wang et al., 2019b) explicitly utilizes edges as additional features and performs MLPs on the edge embeddings to better model local geometric structures. In Liu et al. (2020), they investigate and compare different architecture components of point cloud processing architectures.

Pseudo grid-based methods create artificial convolution kernels to which the points are projected. KPConv (Thomas et al., 2019), constructs a pseudo grid where the grid points are uniformly placed in a ball. Since the pseudo grid is structured, the standard convolution can be applied. Xu et al. (2021) further extends KPConv by making pseudo grid learnable. In their approach, the convolutional kernels are initialized by a random matrix and optimized during training.

On the other hand, some methods predict the convolutional kernels on-the-fly using some features. They are typically called adaptive weight methods. PointCNN (Li et al., 2018) is among the first to achieve convolution on unstructured point clouds. It performs the convolution by first reordering points into a canonical order and then applying depthwise separable convolution (Chollet, 2017). All operations are learnable. PointConv (Wu, Qi, and Fuxin, 2019) dynamically generates a convolution filter for each local group. The relative coordinates relative to the group center are used to produce the filter. Similarly, Liu et al. (2019) produces filters using geometric features such as relative positions, absolute positions, and euclidean distance. A categorization of local aggregation operators are shown in Figure 1.10.

Inspired by the success of Transformers (Vaswani et al., 2017) in the NLP field, many studies have tried to construct transformer-based methods for point cloud data. Here, we focus on transformers that are designed for point cloud segmentation. Since self-attention is a set operator, it can be straightforwardly applied to point clouds. PAT (Yang et al., 2019) applies the original self-attention directly to point clouds. PointASNL (Yan et al., 2020), on the other hand, combines a point convolution with self-attention so that the network considers local and global characteristics simultaneously. Point Transformer (Zhao et al., 2021) applies vector attention to the local subset of points to better model the local structure. Point-BERT (Yu et al., 2022) enhances the performance by masked point modeling. Stratified Transformer (Lai et al., 2022) achieves efficient modeling of long-range dependencies using the stratified strategy. Wang et al. (2019a) uses edges as a descriptor indicating the relationships between a center point and its neighbors. This idea is sometimes combined with the

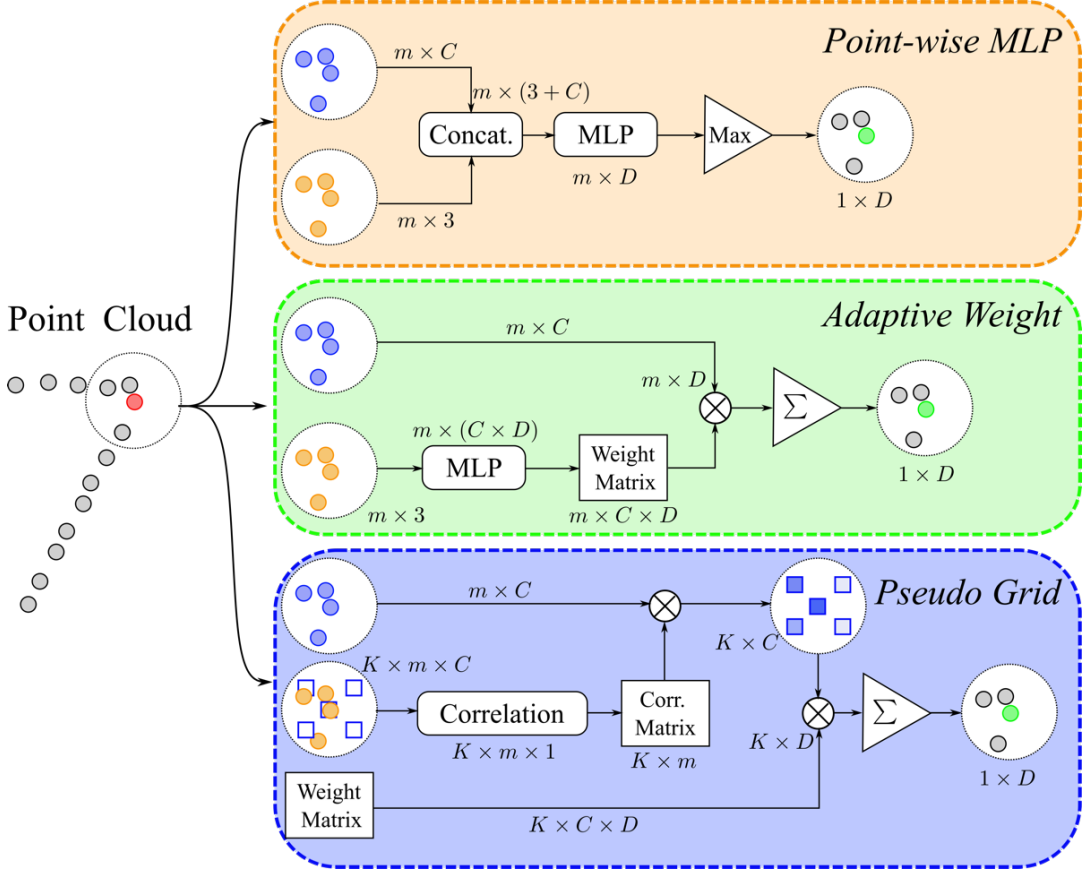


FIGURE 1.10: Categorization of local aggregation operators. m and C denote the number of input points and input channel width, respectively. D denotes the output channel width. “Max” denotes the max-pooling operation. “Concat.” denotes the concatenation operation.

attention mechanism. Zhao et al. (2019) connect all points in the local region and weigh them using vector attention. Xiu et al. (2022a) converts edges into spatial weights in order to model the pairwise relationship between neighboring points.

1.3 Objectives

In this section, we formulate the objectives of this research.

CBD research has been evolving rapidly these days. However, unlike fields like computer vision, there exists no unified standard (e.g., benchmark datasets) to compare the pros and cons of proposed methods in terms of their methodology. This lack of standards results in significant methodological differences in methods. For example, one method may require accurate building footprints whereas the other does not. This is in fact understandable since each event is independent of the others and the data availability and the quality of data are extremely varying. To understand the requirements of various methods, we categorize existing point cloud-based CBD research based on the following standards:

- The use of pre-event data. As described in Section 1.2.1, many image-based methods use pre-event data in conjunction with post-event data to perform change detection. although the use of pre-event data can provide reliable results, it is extremely rare that the pre-event LiDAR point cloud is available in practice. Therefore, the use of pre-event data significantly affects the usability of the methods.
- The use of building footprints/locations. Building footprints are useful in because they can be used for extracting the buildings directly. However, building footprints are rarely available in practice. Although countries like Japan have prepared and made it public the building footprints for each city, they are still unavailable or confidential in many countries. In such cases, the building locations need to be identified by other means (e.g., annotation by human operators (World Bank - ImageCat Inc., 2010a)). Therefore, the use of building footprints/locations is a vital factor that we need to consider.
- The use of hand-crafted features. Using the hand-crafted feature is a direct way to integrate prior knowledge into the design of the methodology. For example, the planarity of point clusters/segments can reveal whether the those points are smooth or not. On the other hand, hand-crafted

features are not optimal since they are designed based on limited prior knowledge. Therefore, hand-crafted features cannot universally cope with an immensely complex reality. Therefore, methods that extensively rely on hand-crafted features may work well in one case and fails in another.

TABLE 1.2: Strengths and weaknesses of point cloud-based CBD methods.

	Change detection	Geometric analysis	Machine learning	Deep learning
w/o pre-event data	✗	✓	✓	✓
w/o building footprints	✓	✗	✓	✓
w/o hand-crafted features	✗	✗	✗	✓

Table 1.2 shows the categorization result. Change detection-based methods must have pre-event data to detect the changes after the event. Since they typically use the whole point clouds as input, building footprints are not required. However, the identification of collapsed buildings still depends on some hand-crafted thresholds and statistics which can be very different in another event. For geometric analysis-based methods, they primarily attempt to detect damaged buildings based on the geometry of post-event data, and thus the pre-event data is not required. However, most of the geometric methods directly take pre-extracted buildings as input. Therefore, building footprints or locations must be available beforehand. Like the change detection method, they also extensively rely on hand-crafted thresholds which require numerous trial and error to be determined. The machine learning methods, on the other hand, do not require pre-event data or building footprints. Similar to the geometric method, machine learning methods also learn the data based on some geometric or radiometric features. Since most methods take the entire points as input, they do not require building footprints to be available. One drawback of machine learning methods is that they depend excessively on the quality of hand-crafted features. In their method, hand-crafted features are first determined and computed from the raw input. Then, the computed features are fed into some classifiers to perform classification. Since the classifiers only receive the computed features, their performance is directly correlated to the quality of the features. In addition, since these methods perform segmentation before classification, some hand-crafted parameters also need to be determined for the segmentation as well.

As can be observed, the existing conventional methods cannot meet the listed conditions simultaneously. On the other hand, as shown in Table 1.2, deep learning methods can meet the listed conditions at the same time. Deep learning methods do not require the existence of pre-event data since they can be used to learn the geometry of point clouds. They also do not require building footprints as input because the task can be formulated as semantic segmentation (Weber and Kané, 2020). As described in Section 1.2.2, one of the greatest advantages of deep learning is that it learns the suitable feature representation from input data directly; therefore, unlike conventional machine learning, it can be applied directly to the raw data and the learned representations are more flexible and powerful since they are optimized through training. Therefore, deep learning is fairly suitable for CBD.

Despite the seemingly attractive characteristics of deep learning, and despite the remarkable progression of deep learning-based point cloud processing, the topic of deep learning-based CBD using 3D point clouds is largely unexplored. Based on our surveys, we believe that the lack of a suitable dataset on which DNNs can be trained prevents the further advancement of deep learning-based CBD using 3D point clouds. Although the problem is easy to solve at a glance, we note that the development of the dataset for point cloud-based CBD is rather challenging because of two reasons: 1) unlike in image processing where object annotations can be created relatively easily, producing ground truth data for CBD is difficult because the earthquakes are rare and building damage needs to be interpreted by domain experts and 2) even if the ground truth is available, the available data may not be in a good form (e.g., GPS coordinates with systematic and human errors), and they must be preprocessed into suitable forms so that it can be used for developing datasets.

In addition, another correlated issue in deep learning-based CBD using 3D point clouds is that there exists no dedicated method that adapts to CBD based on some domain-specific prior knowledge for this task. Just like the prior knowledge embedded in CNN has successfully produced a huge success, we believe that integrating prior knowledge into CBD is necessary to achieve better performance. Such knowledge would be useful especially when the data is relatively scarce and difficult to collect, which is often the case for CBD.

Based on the above analysis, we set the objectives of this research as follows:

- Objective 1: Develop a suitable dataset for point cloud-based CBD and analyze the performance of existing DNNs.

- Objective 2: Develop a dedicated method for CBD by integrating domain-specific prior knowledge into the model design. The resulting method should be able to detect collapsed buildings with different structures, appearances, and collapse causes.

1.4 Contributions and Dissertation Outline

The contributions of this research are summarized as follows:

- A new and reliable dataset for 3D point cloud-based CBD is developed. To the best of our knowledge, we are the first to create the 3D point cloud dataset for CBD and benchmark some mainstream DNNs on the developed dataset. The dataset and the analysis of benchmarking can provide valuable insights to advance 3D point cloud-based CBD.
- A new computation unit called Laplacian Unit (LU). LU selectively enhances the useful discontinuities like damage to buildings while suppressing the less useful ones. Extensive experiments are performed to validate the effect of LU and the effectiveness of its components.
- A new method called Laplacian-Enhanced Neural Network (LE-Net) is proposed to tackle CBD. Extensive analyses are performed to investigate LE-Net concerning its strengths, weakness, efficiency, and robustness.

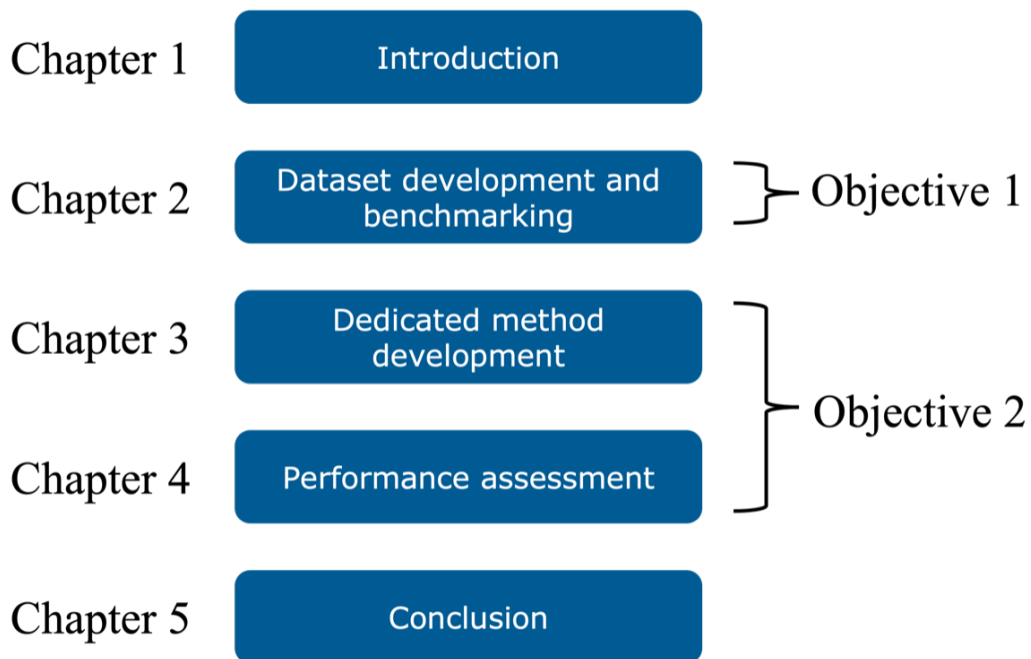


FIGURE 1.11: Dissertation outline

The outline of the dissertation is shown in Figure 1.11.

Chapter 2 presents how the dataset is developed and the analysis regarding benchmarking results. We use the LiDAR and field survey data obtained after the 2016 Kumamoto earthquake to create the dataset, which we call the Mashiki dataset. Then, we benchmark several mainstream DNNs to measure their performance. Among them, we choose PointNet++(Qi et al., 2017b), the standard and widely acknowledged network for point cloud processing, to perform a detailed analysis and reveal that it further needs to be enhanced and adapted to CBD.

Chapter 3 develops LE-Net, a dedicated method for CBD. We first perform an in-depth analysis on the characteristics of building damage. Based on the analysis, we develop LU that can selectively enhance/suppress discontinuities. Using this LU as a core, LE-Net is constructed to tackle CBD.

Chapter 4 performs comprehensive experiments to validate the effectiveness of LE-Net and LU. CBD is performed using the Mashiki dataset and the performance of LE-Net is compared with benchmarking results. Various analysis reveals that LE-Net outperforms other methods significantly. The LU's effect and design choices are also validated. In addition, visual explanation analysis is performed to investigate how LE-Net understands the damage. The efficiency and robustness against density variation are also validated. Subsequently, the adaptability and transferability of LE-Net are validated using the newly created Haiti dataset, which has substantially different characteristics compared to the Mashiki one.

Chapter 5 provides succinct conclusions and discussion of the limitations of this research. Further, some directions for future research are discussed.

Chapter 2

Development of the Mashiki Dataset and Benchmarking

2.1 Introduction

This chapter primarily deals with the problem that there is no suitable dataset for point cloud-based CBD using deep learning. This is a crucial issue concerning deep learning-based research. Deep learning-based research does not advance without a suitable dataset because DNNs are excessively dependant on data. Due to the absence of a dataset, the performance and behavior of DNNs on point cloud-based CBD is still unexplored. Clarifying what DNNs can achieve and cannot achieve is important for providing fruitful directions for future research.

The chapter begins with the introduction of the study area where the dataset is created and the data sources that are used for developing the dataset. Concretely, the field survey data that is used for creating ground truth is described and the LiDAR data is then briefly introduced. Next, the procedure of dataset development is elaborated. After the dataset development, the characteristics of the developed dataset are described using typical examples. Subsequently, the CBD task is formulated and how it can be implemented technically is explained. Then, the benchmarking is performed using the developed dataset. Specifically, this section compares common DNNs for point clouds and investigates their performance. Among those DNNs, we specifically choose PointNet++ as our target of analysis because it is the standard framework for current point cloud processing research. The strengths and limitations of PointNet++ are identified in order to reveal the challenges it faces in CBD and provide fruitful research direction. Lastly, we briefly summarize the findings derived from this chapter.

2.2 Study Area and Data Source

In this section, we introduce the study area of this research. The field survey and LiDAR data used for creating the dataset are subsequently introduced.

2.2.1 Study area

The study area selected in this study is Mashiki Town in Kumamoto Prefecture, Japan. Mashiki Town was severely damaged by the 2016 Kumamoto Earthquakes. Specifically, we chose the central area of Mashiki—which is shown as the green region in Figure 2.1—as the target area to create the dataset. The entire area is about 4km². Buildings in the surveyed regions were densely distributed, with most of them constructed using wooden materials. In terms of appearance, from which DNNs are designed to learn, most buildings adopt non-flat roofs such as gable and hip roofs.

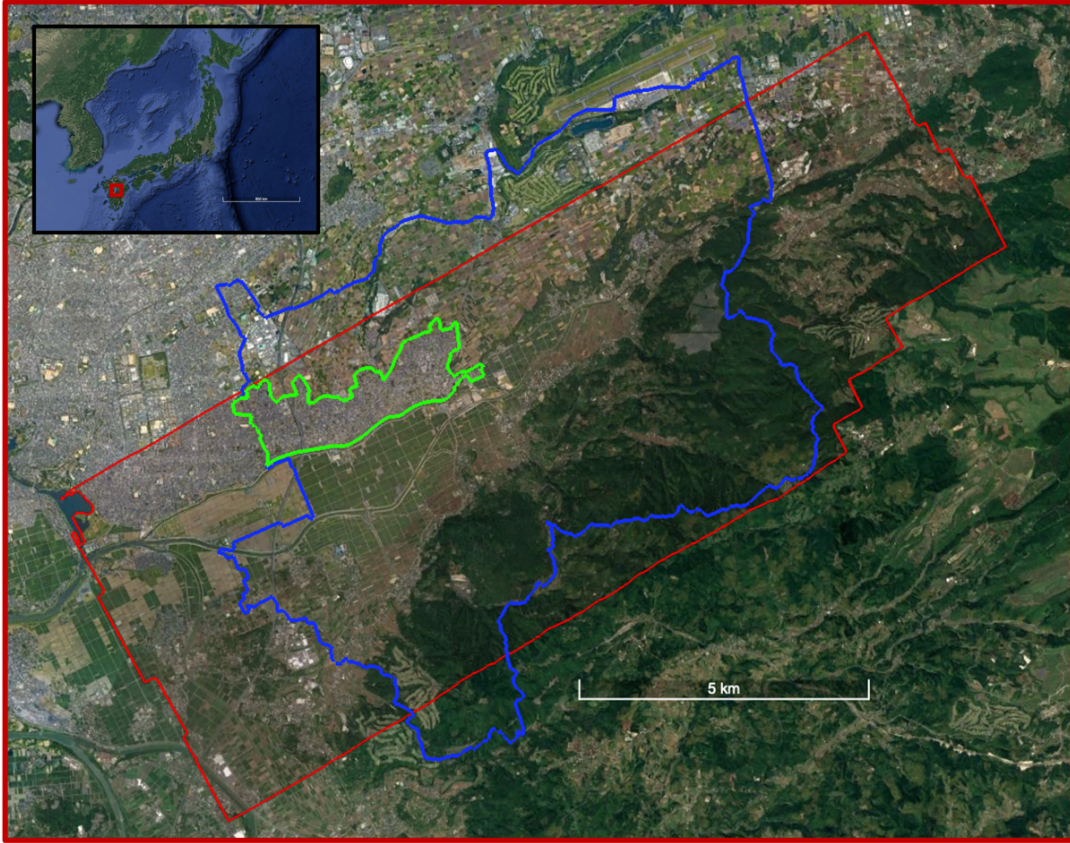


FIGURE 2.1: The study area for developing the Mashiki dataset. Red shaded area: LiDAR survey extent. Blue: the Mashiki Town where surveys are carried out. Green: the central area of Mashiki Town.

We choose Mashiki Town as a study area for two reasons. First, a comprehensive set of data after the 2016 Kumamoto earthquake is available, including

open LiDAR point clouds, aerial photos (aerial surveys and Google Earth historical images), and a detailed field survey result. These reliable data enable a comprehensive analysis of collapsed building detection. Moreover, Mashiki Town is the only place where buildings in LiDAR point clouds can be paired with a unique building ID in the field survey result. The extent of the LiDAR data and study area are shown in Figure 2.1. Second, there are a vast number of collapsed buildings due to the Kumamoto earthquake with various kinds of collapse patterns and causes. This is a desired condition, especially for data-hungry deep-learning models. Therefore, we believe that building models on top of those collapsed buildings can produce a model with rich knowledge of the collapsed buildings. We expect that training the model on the data obtained from Mashiki Town makes the model generalizable to various collapse patterns and causes of wooden buildings.

2.2.2 Field survey

During 14–16 April 2016, a series of earthquakes struck Kumamoto Prefecture, Japan. The foreshock of M_w 6.2 occurred on 14 April, followed by the mainshock of M_w 7.0 on 16 April. After the mainshock, the local governments of Kumamoto Prefecture performed a series of field surveys for assessing the building damage, according to the “operational guideline for damage assessment of residential buildings in disasters” issued by the Cabinet Office of Japan (Cabinet Office of Japan, 2018). If a building was recognized as destroyed (not necessarily collapsed), the type of collapse (i.e., story-collapse or not) was determined by the field investigators (Kawabe et al., 2021). Photographs of the target building were taken by investigators, and the building’s GPS coordinates were recorded. The information collected in the field surveys used in this study included the damage grade, photographs, and GPS coordinates of the surveyed buildings. Damage grades are recorded in the survey sheet.

There are five grades included: “No damage”, “Slight damage”, “Moderate-damage”, “Moderate+” damage, and “Major damage”. These are the damage grades defined by Cabinet Office of Japan (2018). However, these damage grades cannot be used directly because they are defined based on the damage to various parts of a building and the final grade is calculated by accumulating damage scores from all parts (Xiu et al., 2022b). The problem with this calculation is that the major portion of accumulated scores comes from the damage to the building foundation and walls as they are important structural elements in a building. However, the main visible part of a building in the airborne LiDAR point cloud is the roof. Since the roof damage does not correspond well to

the actual damage grade, we need to perform a reclassification of grades such that the resulting grades are suitable for point cloud-based CBD. This will be introduced in the following section.

The simplified recording sheet is shown in Figure 2.2. Field survey results and recorded GPS coordinates are demonstrated in Figure 2.3.

Date				
Owner info.		Building info.		
Investigator info.				
Building address				
Building location map				
Damage grade				
No	Slight	Moderate-	Moderate+	Major

FIGURE 2.2: Example of the (simplified) recording sheet. The damage grade of the target building is marked by a red marker. In this case, the damage grade of the building is “Major damage”.

2.2.3 LiDAR point clouds

In this study, we use the post-mainshock LiDAR dataset acquired on 23 April 2016 (Chiba, [n.d.](#)) by Asia Air Survey Co., Ltd. (Chiba, Japan). It covers the western half of the rupture zone. The airborne survey was conducted, which generated 3–4 points/m² point clouds (Moya et al., [2018](#)). The major visible part of a building is the roof. For brevity, the post-mainshock data are called post-event LiDAR in this study. The point cloud data of the study area are extracted from the original data using the area footprint provided by the Geospatial Information Authority (GSI) of Japan. The visualization of the LiDAR data of the study area is shown in Figure 2.4.

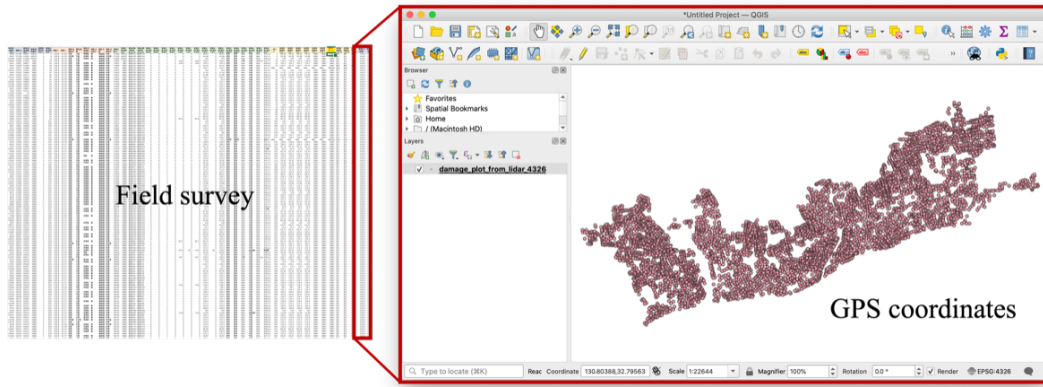


FIGURE 2.3: Field survey results and extracted raw building GPS coordinates (shown in QGIS software).

2.3 Dataset Development Workflow

We design a workflow for developing the dataset. The overview of the workflow is shown in Figure 2.5.

2.3.1 Information extraction

As shown in Figure 2.5, the first step is to extract the damage information. Damage grades and descriptions, as well as the collapse type, are collected during field surveys using recording sheets. An example of the simplified recording sheet is shown in Figure 2.2. These recording sheets are digitized by investigators and stored with photographs taken during the surveys.

Since there are numerous buildings included in the field survey, we need a method that can extract damage grades automatically. The key idea is that the damage grade of the target building is marked in red (see Figure 2.2). Since the positions of the texts indicating damage grades are fixed and the recording sheet is tied to the building's unique ID, the damage grades for each building can be extracted if the red region is detected. To this end, we develop a damage grade extraction algorithm that extracts the red-colored region and assigns buildings their damage grades automatically. The algorithm is shown in Algorithm 1. In essence, it recursively takes a recording sheet image and extracts its damage grade, which is a red-shaded area, using OpenCV (cv2) and NumPy (numpy) libraries in Python programming language. In particular, the red-shaded areas are extracted by first converting the image (the part that contains the damage grade part in the recording sheet) from RGB to HSV space so that the red region can be more easily separated from other regions. The damage grade is recognized by the position of the red-shaded area.

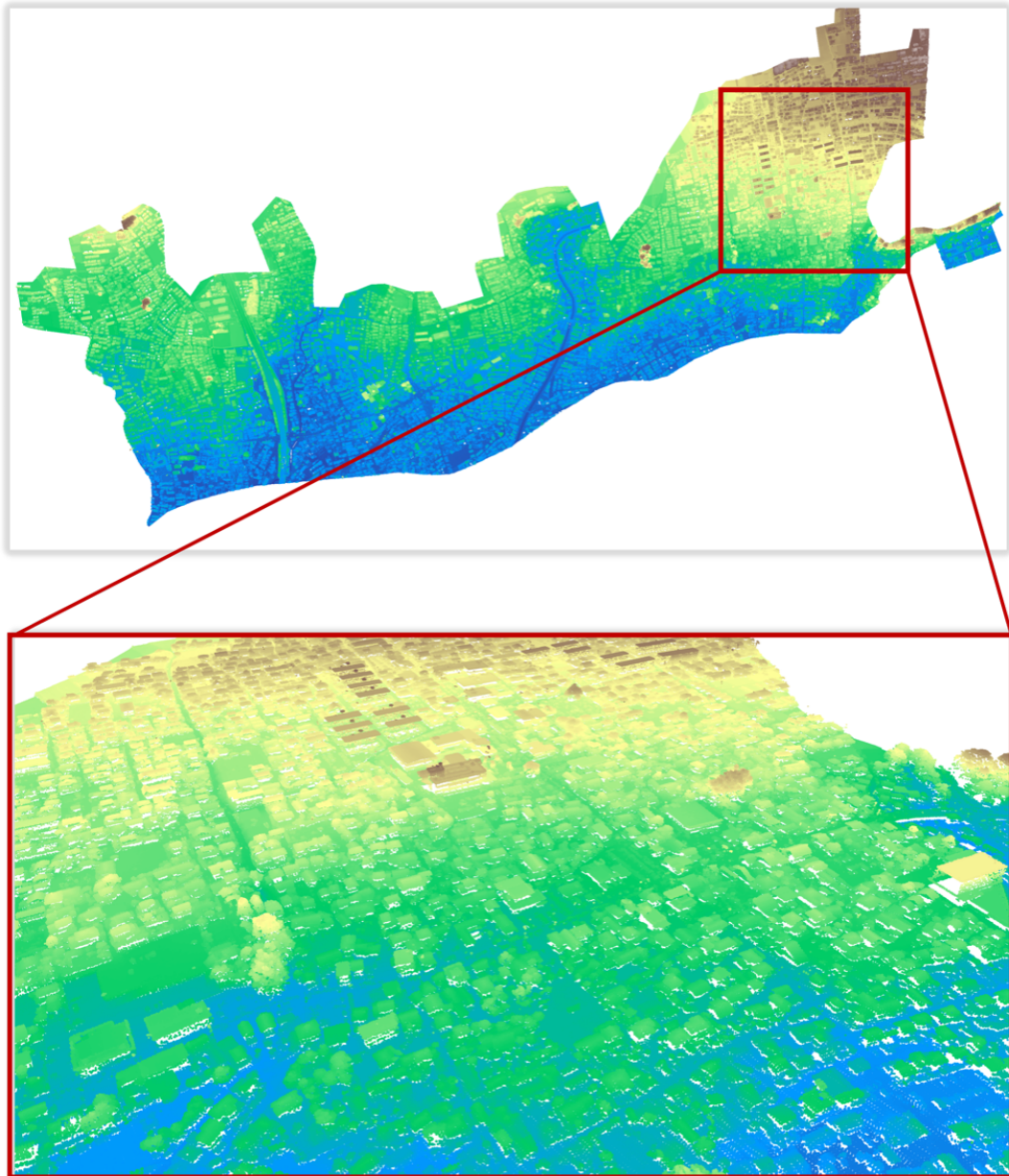


FIGURE 2.4: Visualization of post-event LiDAR data used for developing the Mashiki dataset. Points are colored by their elevation values.

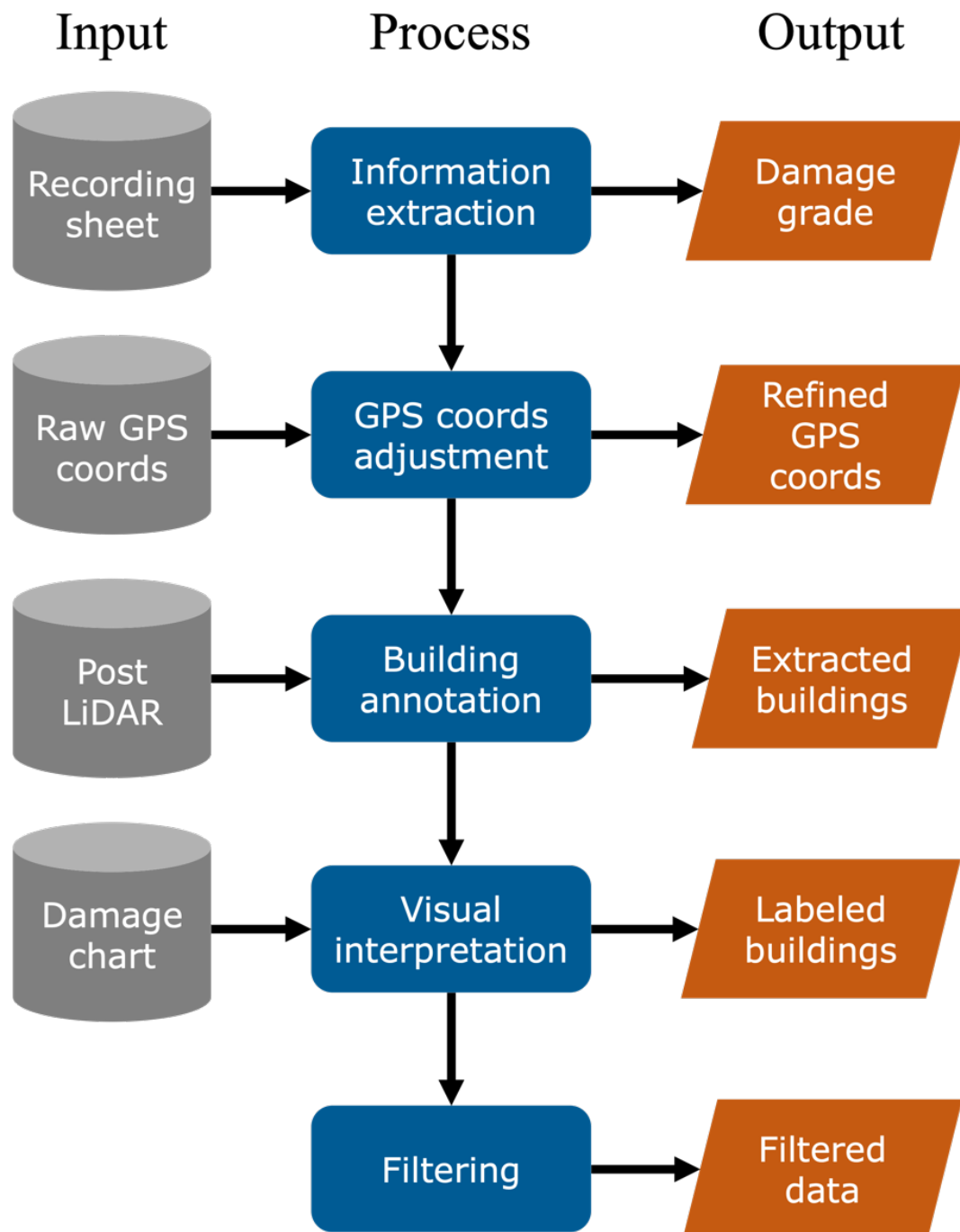


FIGURE 2.5: The Mashiki dataset development workflow.

Algorithm 1 Python-style automatic damage grade extraction from recording sheets.

```

1: import cv2, numpy as np
2:  $n$  = number of buildings
3:  $pos$  = [posNo, posSlight, posMod-, posMod+, posMajor]
4:  $\delta$  = 40
5: DmgGradeList = []
6: for  $n$  iterations do
7:   SheetImage = LoadSheetImage( $ith$ ImagePath)
8:   DmgPart = ExtractDamageGradesPart(SheetImage)
9:   DmgPart = cv2.cvtColor(DmgPart, cv2.COLOR_BGR2HSV)
10:  mask = cv2.inRange(s, np.array([90 -  $\delta$ , 70, 50]), np.array([90 +  $\delta$ , 255,
    255]))
11:  idxH, idxW = np.where(mask==255)
12:   $v$  = np.median(idxW)
13:  DmgGrade = np.argmin(np.abs( $pos$  -  $v$ )) + 1
14:  DmgGradeList.append(DmgGrade)
15: end for

```

2.3.2 GPS coordinate adjustment

The residential buildings in the surveyed areas are densely distributed. The recorded GPS coordinates, therefore, do not always precisely correspond to the correct buildings, because of various factors (e.g., systematic errors or human errors). To eliminate possible erroneous pairings of the buildings and their coordinates, we manually adjust every GPS coordinate. The overview of the procedure is depicted in Figure 2.6. The building point clouds are converted to 2.5D point clouds and visualized along with building GPS coordinates in QGIS. Then, the GPS coordinates are manually moved to the top of the corresponding buildings. In the case of uncertain pairings, we identify the correct buildings by comparing in situ photographs of the buildings, which are tied with GPS coordinates using IDs, and Google Earth historical images.

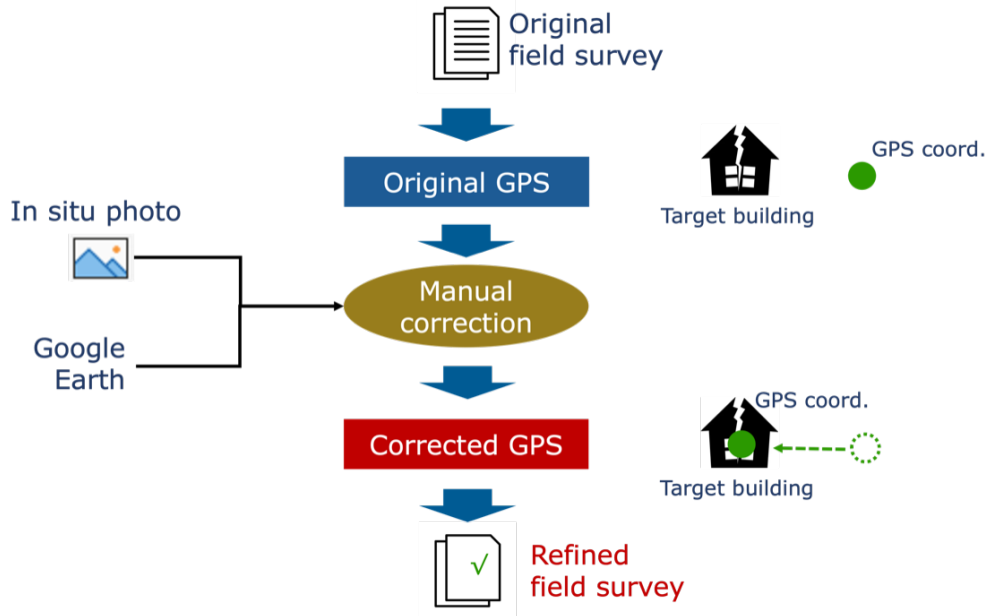


FIGURE 2.6: GPS coordinates adjustment procedure.

2.3.3 Building annotation

The overview of the procedure is shown in Figure 2.8. The adjusted coordinates are used to locate the corresponding buildings in post-event LiDAR point clouds. Specifically, for each coordinate, we use it as the center and extracted all points within a 10m square window (with infinite elevation) out of the whole point clouds to visualize using Python PPTK library. Then, the building part of the point clouds is annotated manually and stored in the NumPy Array format. The boundary ambiguities between buildings and other objects are resolved by

visually assessing backscattered intensities, in situ photos, and Google Earth historical images. Note that the buildings are observed by airborne sensors, and thus, the information of the structures that are below roofs is extremely limited. Furthermore, the debris of collapsed buildings is excluded, in order to minimize possible incorrect annotations. In the case of fully collapsed buildings, we observe that their roof remained either structurally intact or partially deformed, and thus, their roofs are also annotated. In other words, all of the extracted building data consist of either structurally damaged roofs in the case of collapsed buildings or structurally intact roofs for other buildings (these roofs may contain surface damage).

Subsequently, we extract the building point clouds, along with their contextual information, which we call the building patch. An initial bounding box is computed using the annotated building roof, and subsequently, the longer side of the bounding box is chosen as the size of the square bounding box for patch extraction. Finally, the square bounding box is uniformly dilated with a buffer, in order to increase the amount of surroundings information. In this study, the buffer is set as 2 m.

Concerning implementation, we have developed a simple browser-based annotation tool based on Jupyter Notebook for efficient annotation. The overview and the workflow of the annotation tool are shown in Figure 2.7.

2.3.4 Visual interpretation

It is common practice to use visually interpreted building damage using aerial images as reference data to train the DNNs for emergency mapping (Miura, Aridome, and Matsuoka, 2020; Naito et al., 2020). Therefore, we visually interpret the post-event LiDAR dataset, in order to extract collapsed buildings, according to the damage pattern chart proposed by Okada and Takai, 2000. For reference, the damage pattern chart is shown in Figure 2.9. Specifically, the “D5” buildings are classified as “Collapsed” class. Note that we only visually interpret and classify the building point clouds that are initially classified as “Major damage” (not necessarily collapsed) buildings by field investigations into either the “Non-collapsed” or “Collapsed” classes. Subsequently, buildings that are not classified as “Major damage” by the field investigations are labeled as “Non-collapsed”. Then, we assign damage classes to corresponding building roofs in building patches and labeled the remaining points as “Background” (see Figure 2.8).


```

jupyter annotation_v5 (unsaved changes)
File Edit View Insert Cell Kernel Widgets Help Not Trusted random
In [1321]: # no damage
b_1 = v.get('selected')
annotation = make_label_v2(fn, b_1, b_2, b_3, b_4, b_5, b_6, ground, non_ground,
                           high_veg, non_res, annotation, partial=partial, ori_ind=ind)
b_1 = np.asarray([])
np.unique(annotation)
[ 2  2  2 ... 12 12 12]
partial update !
Out[1321]: array([ 1,  2,  5,  6, 11, 12, 13, 14, 15], dtype=uint8)

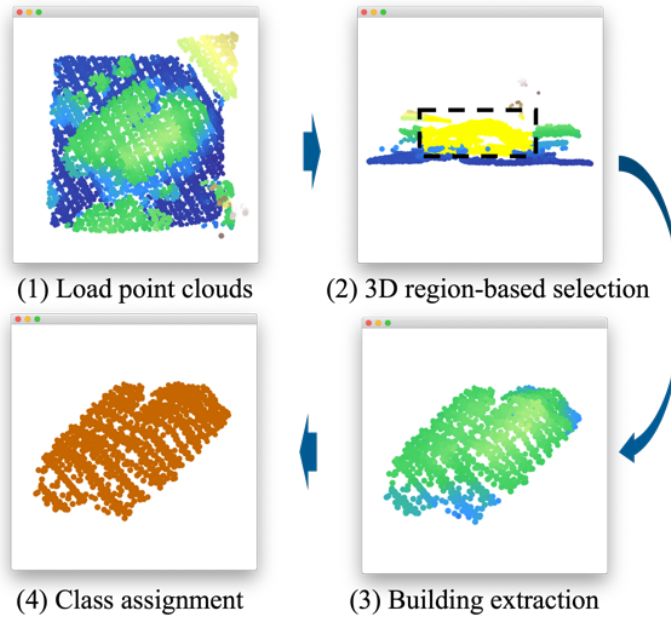
In [1397]: # slight damage
b_2 = v.get('selected')
annotation = make_label_v2(fn, b_1, b_2, b_3, b_4, b_5, b_6, ground, non_ground,
                           high_veg, non_res, annotation, partial=partial, ori_ind=ind)
b_2 = np.asarray([])
np.unique(annotation)
[15 15 15 ...  2  5  5]
partial update !
Out[1397]: array([ 1,  2,  5,  6, 12, 13, 14, 15, 16], dtype=uint8)

In [1608]: # moderate - damage
b_3 = v.get('selected')
annotation = make_label_v2(fn, b_1, b_2, b_3, b_4, b_5, b_6, ground, non_ground,
                           high_veg, non_res, annotation, partial=partial, ori_ind=ind)
np.unique(annotation)
b_3 = np.asarray([])
[ 6  6  6 ...  1  2  2]
partial update !

In [1403]: # moderate + damage
b_4 = v.get('selected')
annotation = make_label_v2(fn, b_1, b_2, b_3, b_4, b_5, b_6, ground, non_ground,
                           high_veg, non_res, annotation, partial=partial, ori_ind=ind)
b_4 = np.asarray([])
np.unique(annotation)
[ 2  5  6 ... 12 12 12]
partial update !
Out[1403]: array([ 1,  2,  5,  6, 12, 13, 14, 15], dtype=uint8)

```

(a) Browser based annotation tool



(b) Annotation workflow

FIGURE 2.7: Annotation tool for developing the Mashiki dataset. (a): Browser based annotation tool. The tool is mainly developed using Jupyter Notebook and PPTK library using Python programming language. Only a fraction of the code is shown in this figure. (b): Annotation workflow. This tool first loads the point cloud of a particular building. Then, it launches the manual selection functionality. During the manual selection, a user can use functions like “rotate”, “translate”, “zoom in” and “zoom out” to choose a suitable position to extract the desired part of the point cloud. Lastly, the class is assigned to the extracted points.

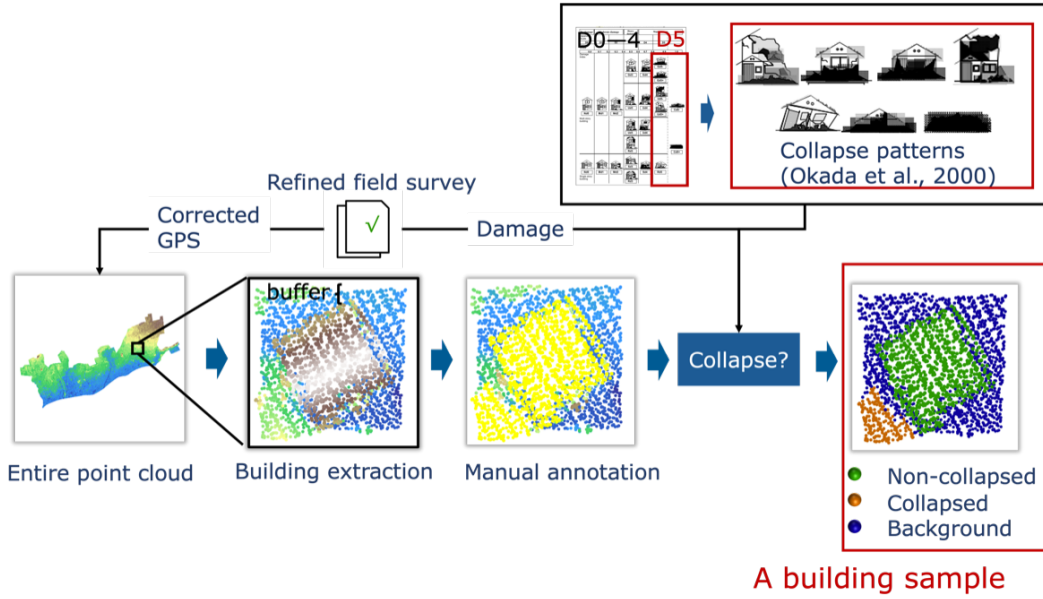


FIGURE 2.8: Building annotation and visual interpretation procedure.

2.3.5 Filtering

The sizes of the buildings are non-uniformly distributed in the extracted buildings. The areas of some buildings are extremely small, comprised of no more than 100 points (e.g., very small houses), whereas some of them are extremely large (e.g., temples) with more than a few thousand points. In order to reduce such outliers, percentile-based filtering is implemented. We first calculate the areas of each building and the numbers of points it contained. Then, for the calculated areas and number of points, we retain the buildings between the first and 99th percentiles. Note that we only retain the buildings that meet both aforementioned conditions.

2.4 Statistics and Examples

TABLE 2.1: Statistics of the Mashiki dataset. #buildings indicate the number of building contained in each class. #points show the total number of points in each class.

	Non-collapsed	collapsed	Background
#buildings	5,609	428	-
#points	3,033,254	331,503	4,912,807

As listed in Table 2.1, there are 6,037 residential buildings in total where most of the buildings belong to the “Non-collapsed” category. Thus, the dataset

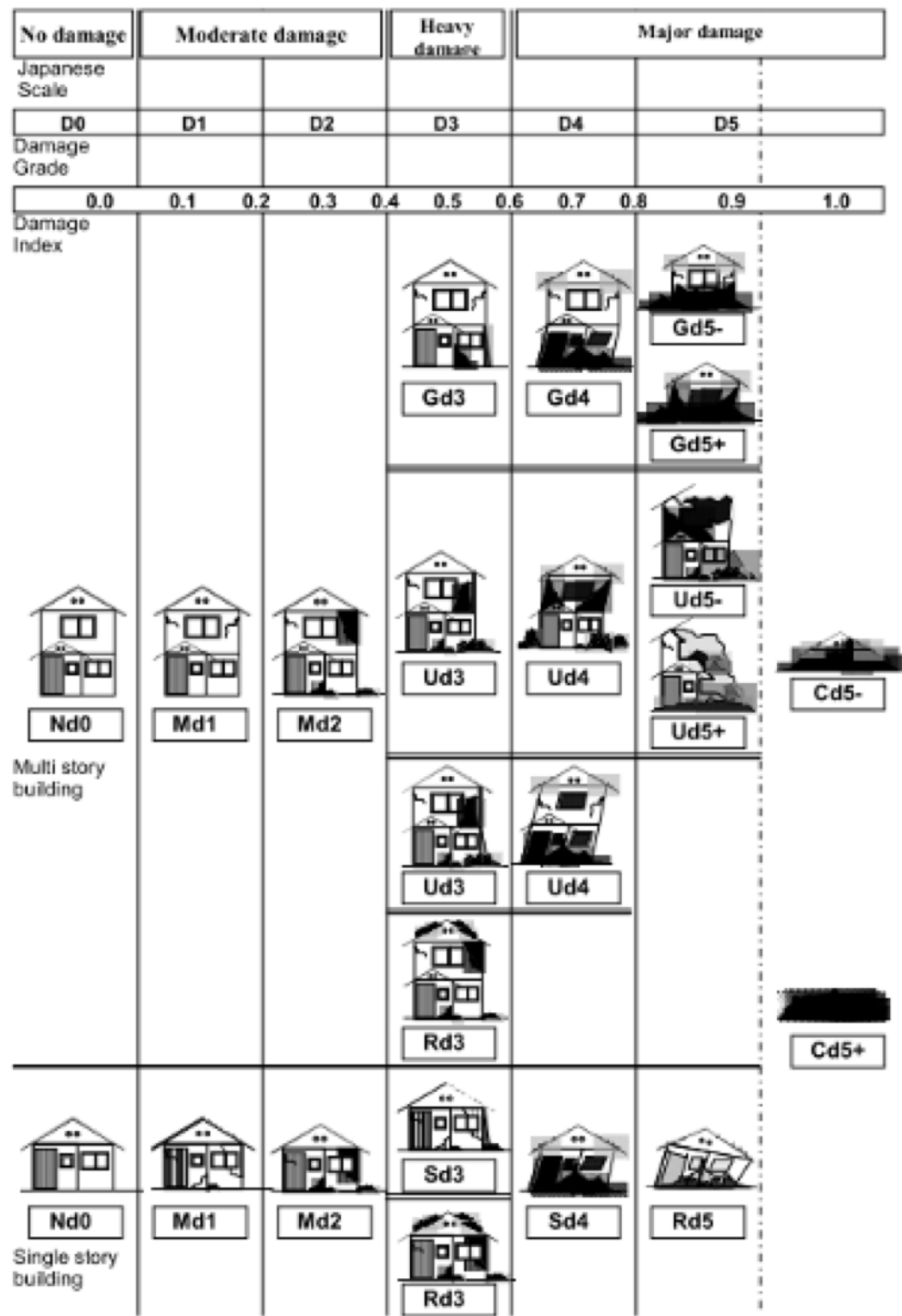


FIGURE 2.9: Damage pattern chart (Okada and Takai, 2000).

TABLE 2.2: #points in training, validation, and test regions.

	Non-collapsed	Collapsed	Background
Training	935,819	152,530	1,587,899
Validation	692,663	71,226	1,131,705
Test	1,404,772	107,747	2,193,203

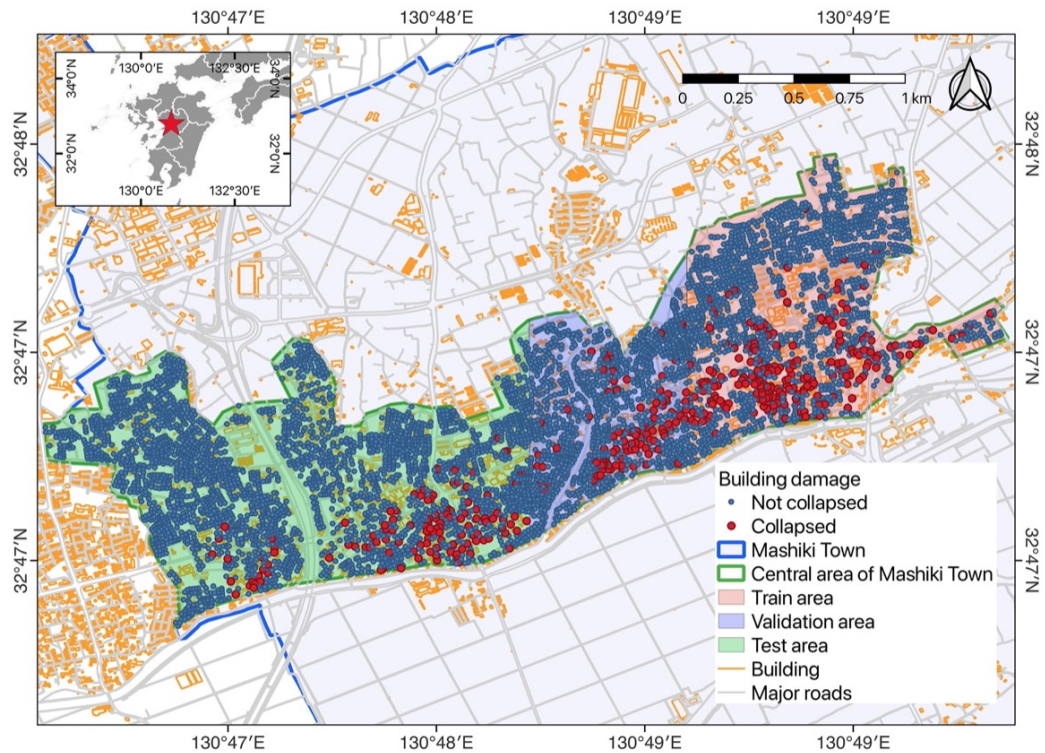


FIGURE 2.10: Distribution of collapsed and non-collapsed buildings in the Mashiki dataset. The entire area is geospatially split into training, validation, and test regions.

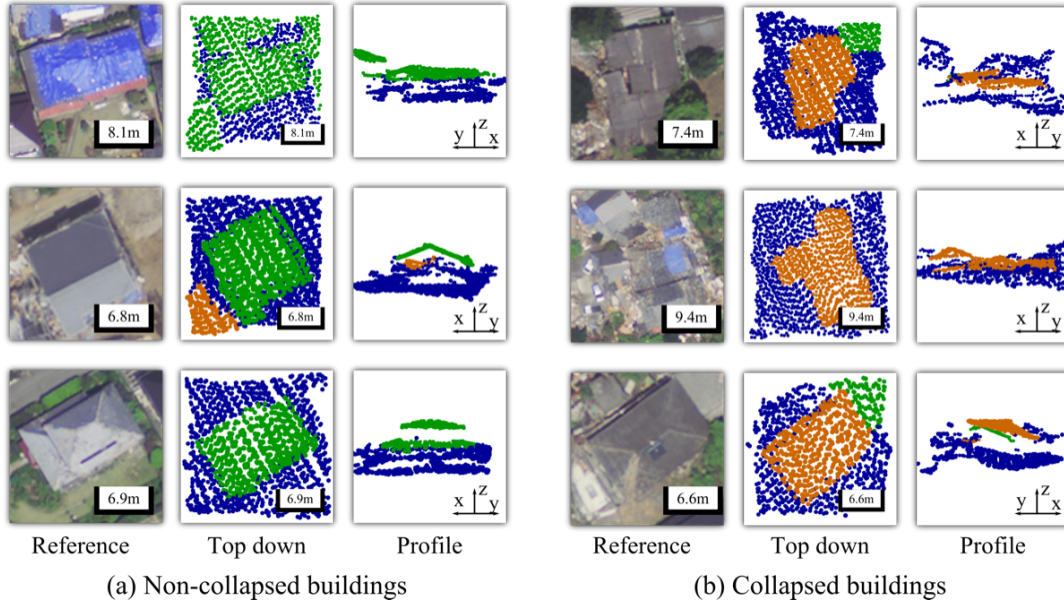


FIGURE 2.11: Examples of collapsed and non-collapsed buildings in the Mashiki dataset. For example, the top row of (a) shows a blue sheet-covered non-collapsed building while the second row of (b) shows a completely collapsed building. Green, brown, and blue points indicate the non-collapsed building, collapsed building, and background points. The “Profile” column shows the profile view of the ground truth point clouds.

is highly imbalanced and some measures need to be taken to combat the class imbalance, which we will explain in more detail in the next section. As shown in Figure 2.10, the entire area is geospatially split into training, validation, and test areas. Approximately 30% of the region is used for training while 20% of them is used for validation. The rest of the region is used for testing. The numbers of points included in each region are reported in Table 2.2.

Several examples of collapsed and non-collapsed buildings are shown in Figure 2.11. A building patch consists of a target building and its surroundings. Surrounding objects include artificial ones such as non-target/non-residential buildings and natural objects like trees. Since the mainly observable part of a building in airborne point clouds is a roof, points that belong to roofs are annotated as building points.

While non-collapsed buildings in aerial images are mainly characterized by vertical views of the roofs, they are often represented in 3D point clouds as a combination of several rigid planes within which points are smoothly distributed. Moreover, the entire roof is often horizontal to the ground plane.

In contrast, collapsed buildings are described by irregular distributions of points. Those points often represent heavily deformed roofs or destroyed building structures, i.e., debris. Debris is usually present near/inside footprints of

collapsed buildings. On the other hand, some collapsed buildings do not undergo heavy roof deformations. An example is shown in the last row of Figure 2.11 (b). As can be observed from the aerial image, the roof of the building appears much healthier than the completely collapsed buildings (the second row of Figure 2.11 (b)); thus, the correct identification of damage from the aerial image is challenging. Furthermore, since the building roof does not collapse to the ground, it is difficult to estimate the damage based on the height difference between the roof and the ground. The limited amount of observable debris further increases the uncertainty. However, this type of collapsed building often undergoes varying levels of inclinations, which are easily observable from 3D point clouds. Consequently, 3D point clouds are more promising than aerial images in detecting a variety of collapsed buildings.

2.5 Task Formulation

In recent studies, damage detection is commonly formulated as a combination of building localization and damage classification (Shen et al., 2021). In this pipeline, building localization in which buildings of interest are detected is followed by damage classification that estimates a category label (i.e., a damage level) for each detected building. This is reasonable because the locations of buildings are often unavailable, as we discussed in Section 1.3. In practice, these two sub-tasks are often combined into a semantic segmentation task (Weber and Kané, 2020). The concept of the task formulation with semantic segmentation is shown in Figure 2.12. In this formulation, the input of the task is a building patch in which a point is represented by its 3D coordinates. The network consumes the input and produces per-point predictions where each point is assigned a category label. As described in Table 2.2 and Figure 2.11, three categories are included: “Collapsed”, “Non-collapsed”, and “Background”. In this case, the performance of the “Background” class reveals how well a model can differentiate buildings from the background, hence showing building localization performance. On the other hand, the performance of the “Collapsed” and “Non-collapsed” classes demonstrate the models’ ability to detect collapsed and non-collapsed buildings. This pipeline does not require pre-event data, building footprints, or hand-crafted features; therefore, we also follow the recent practice and define collapsed building detection as a semantic segmentation task.

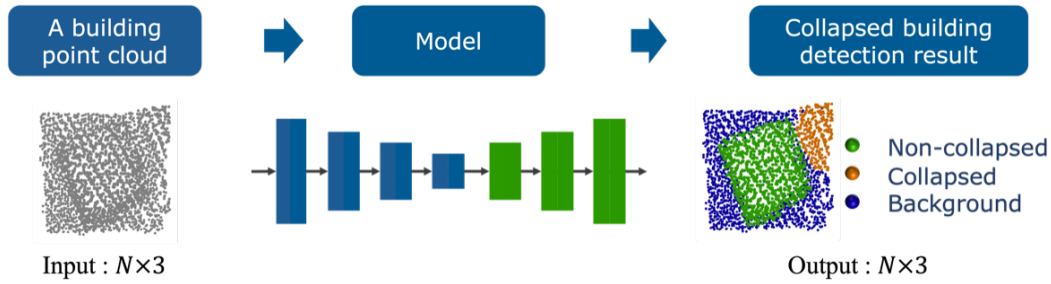


FIGURE 2.12: Task formulation. N represents the number of input points contained in the input building point cloud.

2.6 Benchmarking

In this section, we benchmark some mainstream methods for 3D point cloud deep learning using the Mashiki dataset we just developed. First, the settings for the experiments are described. Next, the evaluation metrics that are used for the experiments are briefly explained. Lastly, we present the results of the experiments. In particular, we focus on investigating the performance of PointNet++, a standard network for point cloud processing, to reveal what can be achieved by PointNet++ and what are the remaining challenges.

2.6.1 Experiment settings

We use building point clouds developed in Section 2.3 as input for the models. For the experiments performed in this study, all available points of the building point clouds are used as input and fed into the model. We use the 3D spatial coordinates as the input feature and the output is per-point 3D vectors that describe the probability of points belonging to certain classes.

During training, we normalize each input point cloud by translating its centroid to the origin. The input is subsequently scaled into a unit ball such that the farthest distance from the origin to a point becomes 1. Random anisotropic scaling in the range of $[0.66, 1.5]$ and random vertical rotation are used for data augmentation. The scaling ensures that the trained models are more robust to scale variations, which are common for buildings. The rotation augmentation can effectively increase the number of samples because the direction that buildings face may vary in practice. After the experiment settings and hyperparameters are determined, both training and validation data are used for training. We use SGD as the optimizer. The total epoch is set to 150. The initial learning rates are set to 0.1. The learning rate is decayed by 0.1 at epochs 90 and 120. The batch size is set to 24.

To combat the class imbalance, we use the inverse class frequency weighted cross entropy as our loss function. We implement all codes using PyTorch deep learning framework. All the experiments are conducted using a machine with an NVIDIA Tesla V100 GPU.

2.6.2 Evaluation metrics

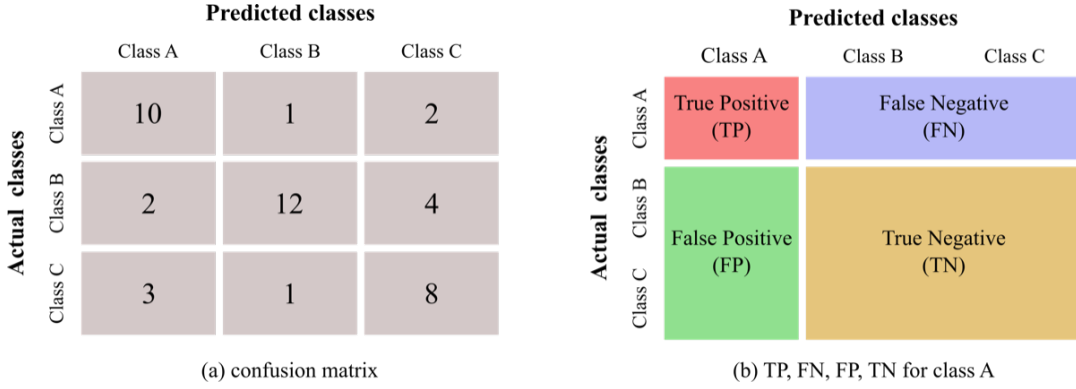


FIGURE 2.13: Confusion matrix calculation. **(a)**: confusion matrix for three-class classification. **(b)**: TP, FN, FP, TN calculation for class A. In this case, $TP = 10$, $FN = 1 + 2 = 3$, $FP = 2 + 3 = 5$, $TN = 12 + 4 + 1 + 8 = 25$.

We adopt several standard evaluation metrics derived from the confusion matrix (Figure 2.13) for the segmentation task to assess the performance. Specifically, we use precision, recall, and Intersection over union as the basic evaluation metrics. They are defined as below:

$$\text{Precision} = \frac{TP}{TP + FN} \quad (2.1)$$

$$\text{Recall} = \frac{TP}{TP + FP} \quad (2.2)$$

$$\text{IoU} = \frac{TP}{TP + FP + FN} \quad (2.3)$$

$$\text{OA} = \frac{TP + TN}{TP + FP + TN + FN} \quad (2.4)$$

In general, precision reflects a classifier's robustness against false alarms whereas recall measures the sensitivity of a classifier to a particular class. IoU, on the other hand, measures the overall performance of a classifier in classifying a particular class. Alternative illustrations of the above metrics are presented in Figure 2.14. Intuitively, precision shows how many samples are actually correctly predicted among the samples that are predicted as a certain class. Therefore, precision is sometimes called "correctness". Recall, on the other hand, shows

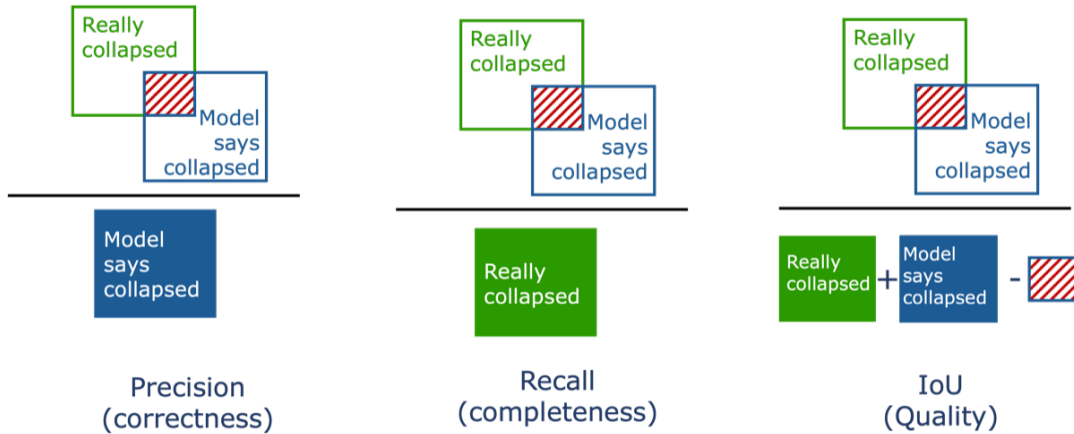


FIGURE 2.14: Illustrations of precision, recall, and IoU.

how many samples are actually found by the models. hence, it is sometimes called “completeness”. As can be seen, the denominator of IoU is a combination of the previous two metrics. therefore, it measures the overall performance of a model. Thus, it is sometimes called “quality”. Mean intersection over union (mIoU), overall accuracy (OA), average precision (mPrecision), and recall (mRecall) are used to evaluate the class-averaged performance of the models. mIoU is a popular metric that is frequently used in semantic segmentation and object detection. It is calculated by averaging class-wise IoUs, thus reflecting an average performance of a classifier over classes. Similarly, mPrecision and mRecall measure certain aspects of the class-averaged performance of a model. On the other hand, OA simply measures the ratio of correct samples over all samples without considering class information.

2.6.3 Results

TABLE 2.3: The benchmarking result of the mainstream deep learning models (%). The bold texts indicate the best performance.

Method	OA	mPrec.	mRec.	mIoU
PointNet	90.6	80.5	84.0	71.5
PointNet++	92.3	81.4	83.7	72.8
DGCNN	90.3	82.6	82.8	72.0

The result is shown in Table 2.3. We compare the performance of PointNet (Qi et al., 2017a), PointNet++ (Qi et al., 2017b), and DGCNN (Wang et al., 2019b), as they are widely acknowledged methods for 3D point cloud deep learning. As can be observed, PointNet++ outperforms the others both in

terms of OA and mIoU. Since OA measures the performance when class imbalance is considered whereas mIoU measures the overall performance when class imbalance is considered, PointNet++ is a better model concerning the overall performance. On the other hand, PointNet achieves the best score in terms of mRecall whereas DGCNN achieves the best score in mPrecision. Therefore, PointNet is better at finding the target buildings and DGCNN has a better ability to suppress false alarms. PointNet++ shows better robustness under metrics that measure the performance of the specific aspects of models; hence, it is a more generic model compared to the others.

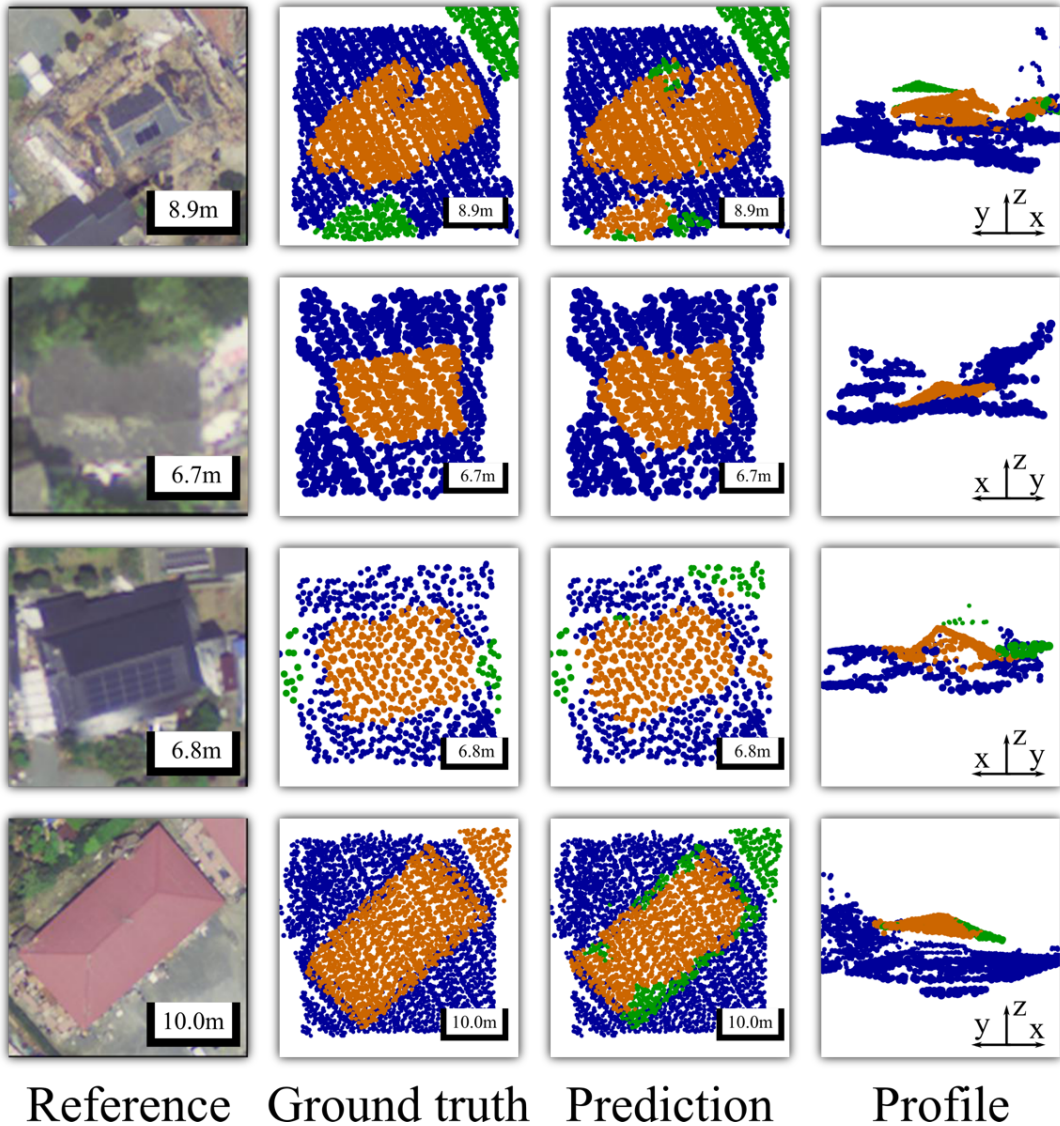


FIGURE 2.15: Result of PointNet++ on the Mashiki dataset. From the top to the bottom row: a collapsed building with a strongly deformed roof; a totally collapsed building; a collapsed building without strong damage to the roof; a building with an inclination.

To better understand what kind of collapsed buildings can be detected, we perform qualitative analysis on PointNet++. The result is shown in Figure 2.15. The top row shows a building with a strong damaged roof. As can be seen, the prediction made by PointNet++ is quite accurate compared with the ground truth. We think that the deformation of the roof presents a distinct pattern that is quite different from that of the non-collapsed buildings, thus making it relatively easier to be detected. Also, although the building suffers from a ground floor collapse, its roof is still above the ground. Therefore, its remaining height may also help to separate the building from the ground. The second row shows a totally collapsed building. As can be seen from the “Profile” column, its roof touches the ground with much debris around. In such a case, PointNet++ still is able to make good predictions. However, we observe a slight dilation of prediction compared with the ground truth. We believe that it is caused by the ambiguity between destroyed roofs and debris since the roof touches the ground, making it challenging to separate them well. On the other hand, the third row shows a case that a building that suffers from a ground-floor collapse while its roof remains relatively intact. In this case, PointNet++ generates an acceptable prediction with few misclassifications. Although the building roof touches the ground, we conjecture that the relatively complete shape of the roof makes the recognition of the boundary between the roof and the ground easier compared with the second row. In the bottom row, the roof of the target building is nearly not damaged while it presents a significant inclination due to structural failure. As the roof deformation is not severe and the building is still standing, we can observe that PointNet++ shows a few misclassifications. However, since a majority of points are classified correctly, it is proven that PointNet++ can detect this kind of collapsed building with acceptable accuracy.

As shown in Figure 2.15, PointNet++ can recognize various collapsed buildings well. However, it shows some limitations in recognizing some challenging collapsed buildings. The examples are shown in Figure 2.16. The first row shows a case where the building is inclined without heavy damage on the roof. Moreover, although it presents an inclination, it is still standing. In this case, although PointNet++ predicts half of the points well, it still makes misclassifications by making incorrect classifications for the rest. We speculate that the relatively intact shape of the roof and its remaining height above the ground confuses PointNet++. Besides, this example shows that PointNet++ cannot recognize the collapsed building solely based on the significant inclinations. In the second row, the roof of the target building is nearly undamaged. Further,

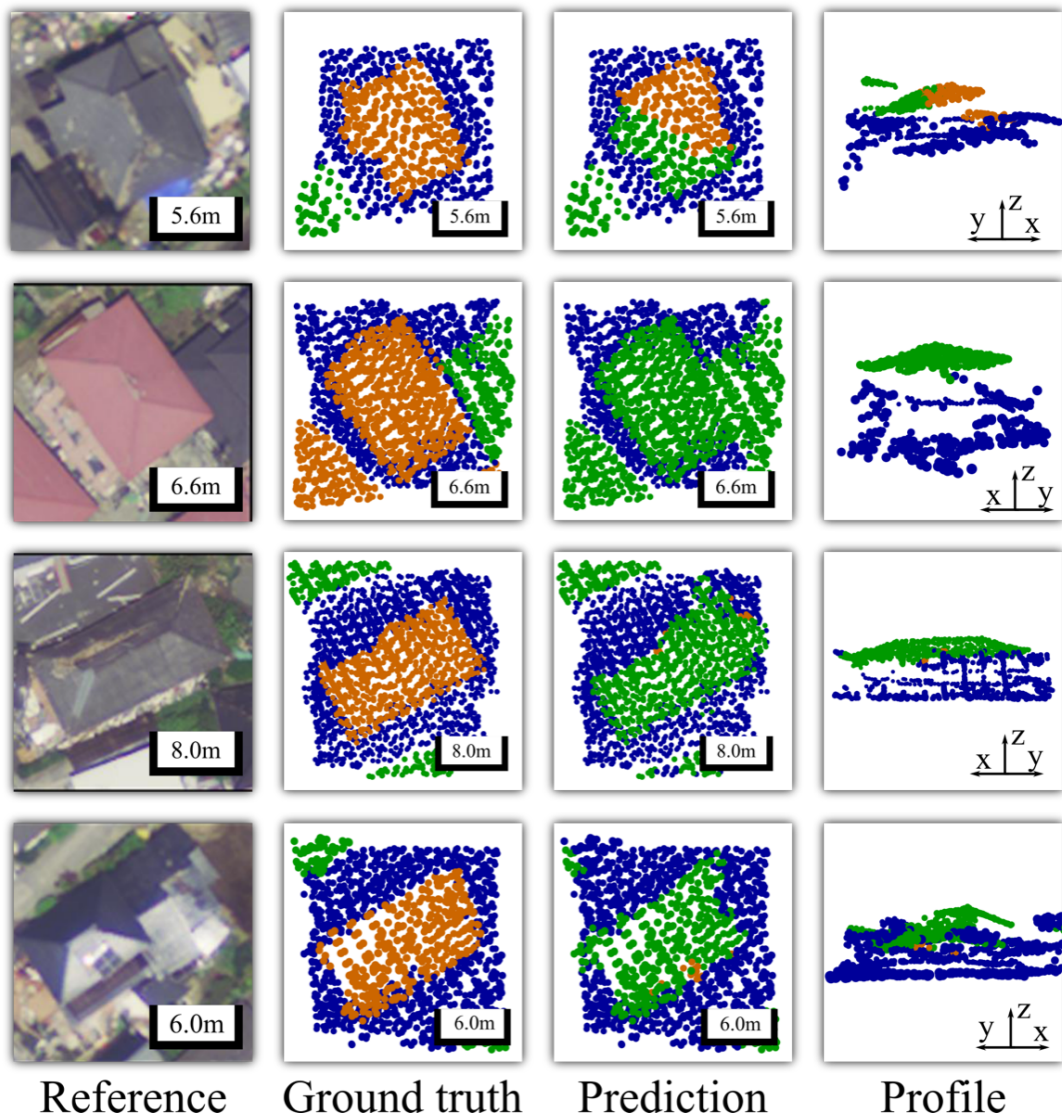


FIGURE 2.16: Failure modes of PointNet++ on the Mashiki dataset.

the building presents only a slight inclination. In this case, PointNet++ completely fails at recognizing this building. We believe that the degree of damage is very challenging to be inferred from the building alone. In the third row, the target building suffers a large crack on the roof. However, as can be seen from the “Profile” column, the building does not show a significant inclination and a large crack is not manifested as a significant deformation as well. Therefore, PointNet++ misses the building entirely, demonstrating its lack of capability to detect slight deformations. In the last two, we observe that the target collapsed building’s roof is nearly intact and its inclination is not severe. As a result, PointNet++ fails completely in this case. We speculate that the lack of significant signatures of damage on the building itself makes the detection extremely challenging.

Based on the above results and analysis, it is still challenging for PointNet++, the standard network for point cloud analysis, to detect all types of collapsed buildings well. The prediction errors are significant especially when the building itself does not show significant clues of damage. Therefore, we believe that it is necessary to improve PointNet++ by deliberately enhancing the feature representation of damage. In the next chapter, we explore how to achieve this goal.

2.7 Summary

In summary, the Mashiki dataset is developed using the field survey conducted by local governments and LiDAR data obtained after the 2016 Kumamoto earthquake. To create a comprehensive and reliable dataset on which DNNs can be trained, we design a workflow for creating the dataset. The resulting dataset is used for benchmarking various existing methods to investigate their performance. As a result, we find that PointNet++, the standard framework of 3D point cloud processing, works well in terms of general performance. Further quantitative and qualitative analyses of PointNet++ reveal that although it can detect most of the buildings that are severely damaged, it fails when the building itself does not show significant clues of damage. Consequently, we conclude that further effort should be spent to enhance the feature representation of damage such that the detection of collapsed buildings can be improved.

Chapter 3

Development of Laplacian-Enhanced Neural Network (LE-Net)

3.1 Introduction

This chapter introduces Laplacian-Enhanced Neural Network (LE-Net), a DNN dedicated to CBD. Based on the analysis of benchmarking results in Chapter 2, PointNet++ (Qi et al., 2017b), which is the standard network of point cloud processing, needs to be further enhanced and adapted to the CBD scenario. In particular, we observe that the buildings that do not show significant clues of damage are difficult to be detected. In other words, we need to make the feature representation of damage more salient so that those buildings become detectable. To do this, prior knowledge of damage is required so that the feature representation of the damaged part can be emphasized by incorporating the knowledge into the model design.

However, there are two issues we need to deal with:

- What kind of prior knowledge is useful in detecting collapsed buildings?
- How can we integrate this prior knowledge into the actual model design?

The first question is undoubtedly an open question. Despite that there might be a lot of prior knowledge that may be useful, an obvious way is to consider the general nature of building damage. This general nature of building needs to be the general characteristic of collapsed buildings and it must be manifested as certain forms in 3D point clouds so that DNNs can learn it.

The second question, in contrast, is more related to the nature of point cloud DNNs. The consideration of this question is vital because our idea/prior knowledge must work in practice in order to improve collapsed buildings. Since

our input to the model is 3D coordinates of points, DNNs learn the surface geometry of point clouds. From this point of view, it is clear that our idea must be related to the geometry of collapsed buildings. Therefore, we need to find prior knowledge that can be represented as a geometric pattern. In addition, this pattern needs to be simple in terms of computation. This is because DNNs are by default optimized through numerous iterations; hence, the computation of each operation must be very efficient.

In this research, the answer we found is to use *smoothness* as a tool to represent building damage. The smoothness is a local property that describes how a point is different from its neighbors. Since building damage in many cases is described by local non-smooth and deformed patterns, smoothness can be a useful concept to represent damage.

To numerically encode smoothness while being very efficient to compute, we adopt the discrete Laplacian as our basic tool to encode smoothness. The discrete Laplacian encodes how different the center point is from its neighbors and is easy to compute, which makes it very fitting for our purpose. Furthermore, our analysis verifies that the discrete Laplacian in conjunction with the hierarchical learning framework can detect inclinations, which is a global property. Thus, the discrete Laplacian is suitable for improving DNNs. However, one vital issue is that the position of the target building is unknown. Therefore, the discrete Laplacian needs to be computed for each point in the input point clouds. Since the input point cloud is a building patch that contains buildings and their surroundings, any damage-like pattern (e.g., tree crowns) may appear in the scene and confuse the model.

To resolve the above issue, we propose Laplacian Unit (LU) that enhances the discrete Laplacian by giving it the ability to enhance/suppress the discrete Laplacian selectively. Moreover, the discrete Laplacian is further enhanced by adding a residual connection, which enriches it using a global feature and eases optimization simultaneously.

Using LUs as core building blocks, we construct LE-Net, which is a DNN specifically designed for CBD. The design of LE-Net generally follows PointNet++; however, unlike PointNet++, we apply LU at each level of the hierarchy so that the damaged part is progressively enhanced along the hierarchy.

This chapter first performs an in-depth analysis concerning building damage and subsequently introduces the tool, the discrete Laplacian, that can enhance the damage representation. Then, the design of LU is introduced along with its rationale. Lastly, we elaborate on how LE-Net is constructed.

3.2 How to Represent Damage Numerically

In this section, we investigate how the damage can be represented in a form that is suitable for DNNs. This section begins with an analysis of the nature of building damage. Based on this analysis, we subsequently propose a way to numerically represent damage and verify its usefulness as well as limitations.

3.2.1 Analysis of damage characteristics

In essence, collapsed building detection amounts to learning the pattern of damage in order to separate collapsed buildings from non-collapsed ones. DNNs essentially learn from the geometry of the point clouds. Therefore, the distinction between non-collapsed and collapsed buildings must be learnable from their geometries. Although letting DNNs learn such knowledge of buildings from scratch is possible, we believe that integrating prior knowledge about collapsed buildings is beneficial to the learning of building damage. To achieve this, however, we need to represent damage in a form that is generally applicable to every collapse form. The knowledge that is only useful in some situations and useless or even harmful otherwise may be discarded by DNNs during the learning process since it is unreliable.

Fortunately, damage to buildings has a simple but general pattern. Figure 3.1 shows photos of non-collapsed and collapsed buildings. As can be seen from the figure, the non-collapsed building is characterized by rigid and smooth patterns whereas the non-collapsed one is described by deformed and non-smooth patterns. The deformed and non-smooth patterns are typically observable in collapsed buildings. For example, we can often observe debris that represents destroyed building elements when a building collapses.

On the other hand, we must also verify that this observation holds true for 3D point clouds. According to Figure 3.1, it can be observed qualitatively that point clouds also demonstrate similar patterns as photographs. Concretely, deformed and non-smooth patterns are described as irregular point distributions, whereas non-collapsed buildings are characterized by smoothly distributed points. Therefore, we believe that this prior knowledge about smoothness is simple but general knowledge that helps DNNs to learn about building damage better.

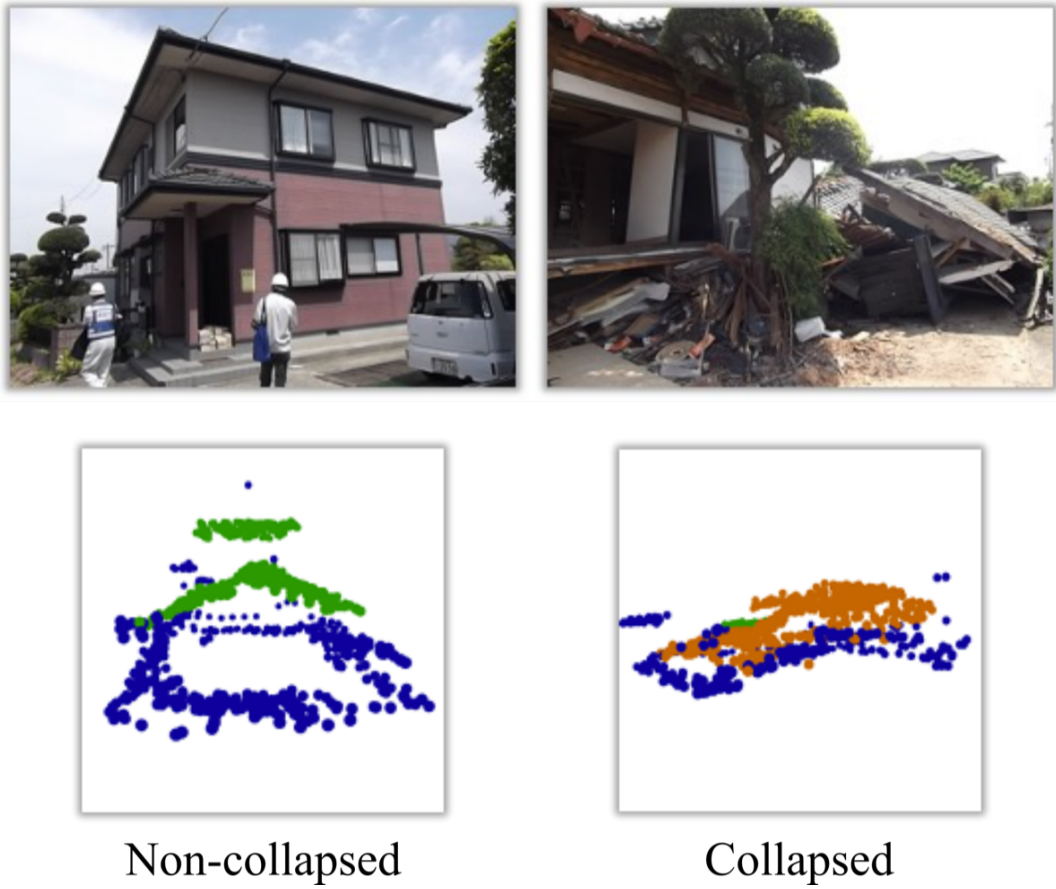


FIGURE 3.1: Comparison of non-collapsed and collapsed buildings. Top: photographs of non-collapsed and collapsed buildings. Bottom: point clouds of non-collapsed and collapsed buildings.

3.2.2 Discrete Laplacian

The question is, how to describe the damage characteristics or smoothness numerically so that it can be integrated into the DNN model design? There might be many choices for encoding smoothness. For example, we can resort to surface normal vectors and compute their local variations for each point to represent smoothness. However, the computation of normal vectors typically requires covariance analysis to be performed on each point, which is computationally expensive when there are many points in the point cloud. In addition, since current DNNs mostly take the form of hierarchical learning, the operator likely needs to be evaluated in each scale of the hierarchy. Therefore, we must choose a tool that can enhance damaged parts while being computationally efficient so that DNNs can compute them on-the-fly.

To this end, we choose the discrete Laplacian (Taubin, 1995) to numerically represent building damage. The discrete Laplacian is an operator that frequently appears in fields such as computer graphics (Desbrun et al., 1999; Sorkine, 2006) or graph processing (Kalofolias, 2016; Dong et al., 2016). The concept of the discrete Laplacian is illustrated in Figure 3.2. Formally, let $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^n \in \mathbb{R}^{n \times d}$ denote the features of a point cloud of some layer in a DNN, where n is the total number of points, d is the feature dimension, and i indexes the points. The discrete Laplacian is defined as:

$$\Delta \mathbf{x}_i = \frac{1}{|\mathcal{N}(i)|} \sum_{j \in \mathcal{N}(i)} (\mathbf{x}_j - \mathbf{x}_i), \quad (3.1)$$

where $\mathcal{N}(i)$ denotes the set of nearest neighbors of point i in space and $|\mathcal{N}(i)|$ represents the total number of neighbors. Equation (3.1) aggregates the difference between a center point and its neighbors. Therefore, the discrete Laplacian describes how different the center point is from its neighbors. In other words, it indicates the smoothness of a region around the center point. As can be seen, the discrete Laplacian is simple to compute and can be easily parallelized; therefore, it is fitting to integrate it into the model design.

3.2.3 Analysis of the discrete Laplacian concerning typical damage

To validate that the discrete Laplacian can be used in a DNN-based hierarchical framework, we visualize the distribution of the discrete Laplacians for non-collapsed buildings and typical collapsed buildings with increasing scales. The result is shown in Figure 3.3. The first row shows a case of a non-collapsed

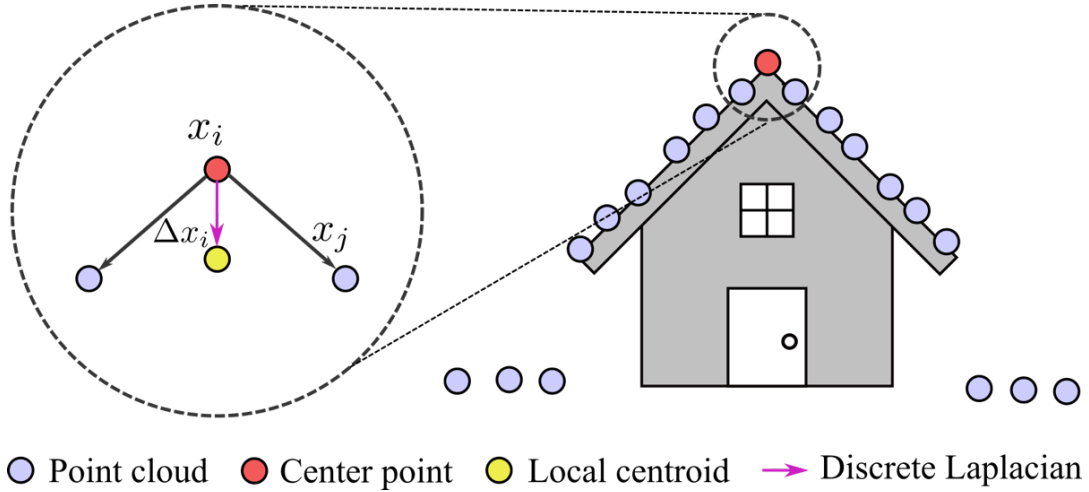


FIGURE 3.2: The illustration of the discrete Laplacian ($\Delta \mathbf{x}_i$ in Equation (3.1)) for a center point i .

building. As we can see, high responses can be observed on the edges of the roofs, forming contours around the roofs. The contour patterns are regular because the roofs are rigid and undamaged. By varying the scale of the discrete Laplacian, it can be observed that the original responses on the contours basically become stronger with increasing scales. The reason is that the non-collapsed building is well-separated from surrounding objects, making its discrete Laplacian pattern and its growth pattern stable.

On the other hand, collapsed buildings present different patterns. For instance, the second row shows a case in which the target building is totally collapsed accompanied by strongly deformed roofs. In such a case, unlike the non-collapsed case, the contours formed by strong responses appear irregular. Concretely, the part where the roofs are still above the ground shows strong responses while the part that touches the ground shows low responses because the part is close to the ground, making it appear smoother. With increasing scale, moreover, some prominent points become identifiable. These points represent deformations of the roofs. Their responses get stronger because the increasing scale increases the inclusion of non-smooth surfaces into the computation of the discrete Laplacian.

The bottom row shows a collapsed building with a significant inclination but without much roof damage. As can be seen, initially the contour formed by strong responses delineates the roof well. However, we observe that the strength of response grows differently with the increasing scale. In particular, the northeast portion of the points grows faster than the southwest portion of the points. This is because the northeast portion of points is higher than the other due to the inclination. The lower part of the points includes more and

more other objects with low heights (e.g., ground), and hence their discrete Laplacian grows slower than the higher part of the buildings.

In general, we find that the smoothness information encoded by the discrete Laplacian is useful based on the following observations. For a non-collapsed building, high smoothness values can be observed on the edges of the roof. For a collapsed building, a decrease in smoothness is observed when the building collapses and the roof touches the ground or other objects. On the other hand, an increase in smoothness is observed when the roof deforms or breaks. Therefore, such changes in smoothness in conjunction with the change of the spatial pattern, i.e., from regular smoothness distribution to irregular one, can be used to distinguish collapsed buildings from non-collapsed ones. Therefore, we believe that the discrete Laplacian is a suitable tool that can identify building damage.

However, the problem with the discrete Laplacian is that it responds to all discontinuities. The discrete Laplacian can represent building damage by encoding smoothness, however, the damage must be distinguished from damage-like patterns. This is because the location of the building is unknown, and thus we cannot straightforwardly determine the damage state of the buildings. As can be observed from the “w/ BG” column of Figure 3.3, the discrete Laplacian shows strong responses on other objects as well. For instance, in the third row, strong responses are observed on vegetation. This is reasonable since tree crowns present irregular point distributions. Mixed with those non-building patterns, it is challenging for DNNs to rely merely on the raw discrete Laplacian. Therefore, we need additional functionality that can transform those values selectively so that only the important one is enhanced/preserved whereas the other gets suppressed. We shall resolve this issue in the following section.

3.3 Laplacian Unit

Despite its usefulness in describing building damage, the discrete Laplacian must be further customized so that it can differentiate useful features from others. In other words, we need to make the operator more flexible such that it can intelligently emphasize useful features in preference to other similar objects. There might be many hand-crafted rules we can rely on to create a certain form of functions, however, a fixed-form function cannot achieve the above objective due to the massive variations of real-world scenes. To resolve this issue, we must create a function that learns to adapt to the incoming data.

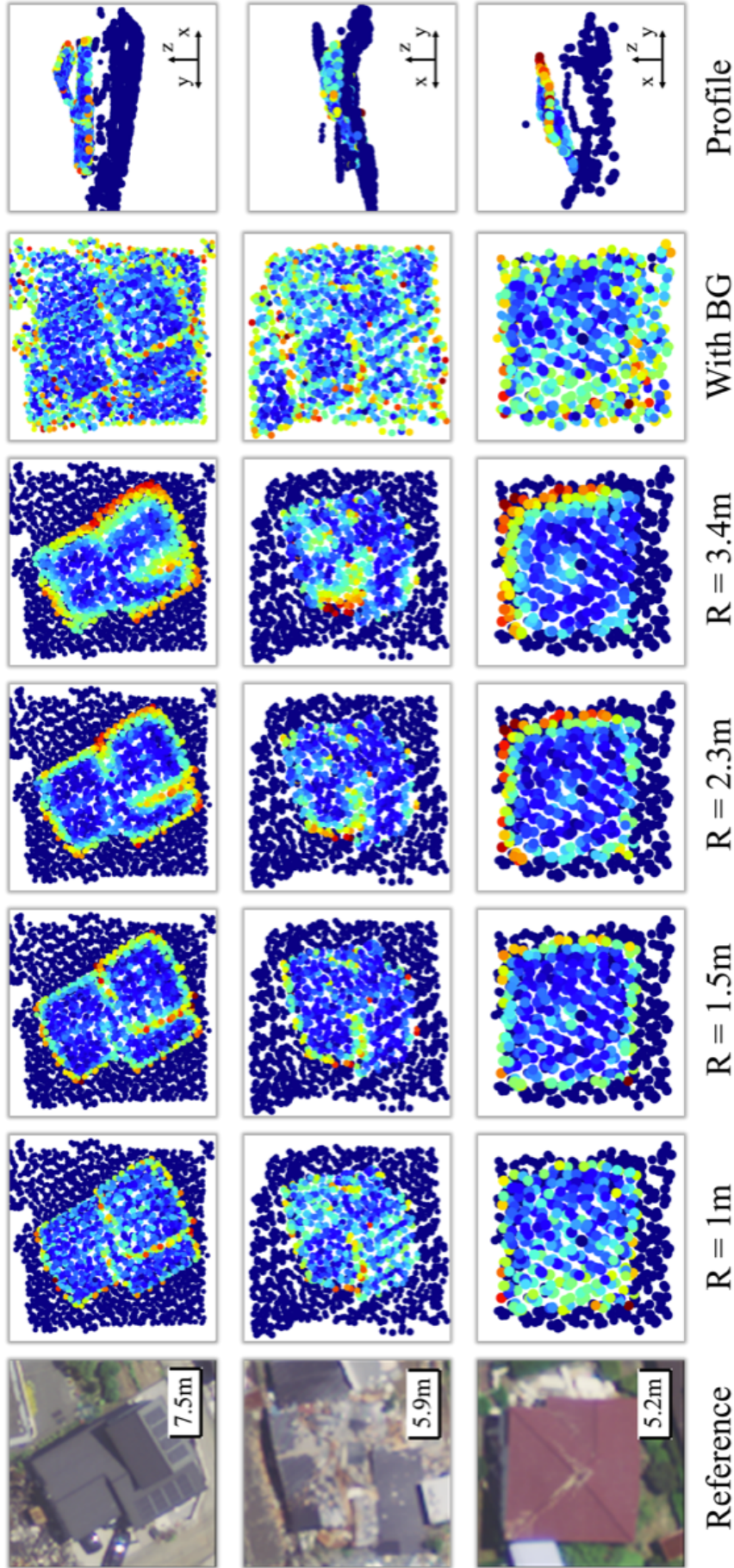


FIGURE 3.3: The comparison of the distribution of discrete Laplacians ($\Delta \mathbf{x}_i$ in Equation 3.1) between collapsed and non-collapsed buildings. Top: the discrete Laplacian of a non-collapsed building. Middle: the discrete Laplacian of a totally collapsed building. Bottom: the discrete Laplacian of a collapsed building with an inclination. The values of background are turned on for the “w/ BG” column whereas they are turned off (zeroed) for other columns for clarity. A bright color indicates a high value of $\|\Delta \mathbf{x}_i\|$, and vice versa, where $\|\cdot\|$ denotes the Euclidean norm. Therefore, bright points indicate non-smooth (big-difference) regions whereas dark points indicate smooth (small-difference) ones. The values of background points (e.g., ground) are zeroed for clarity.

Another vital issue is how to integrate the discrete Laplacian into the current deep learning-based point cloud analysis. In this section, we provide a solution to the above issues and propose the Laplacian Unit (**LU**), a computational module that selectively enhances the feature representation of the useful discontinuities while suppressing the less useful ones.

3.3.1 Disentangling convolution

As the Laplacian is essentially an aggregated local pairwise difference, it should naturally exist in the convolution. Therefore, consider a simple convolution equation:

$$\mathbf{x}'_i = \sum_{j \in \mathcal{N}(i)} w_{ij} \cdot \mathbf{x}_j, \quad (3.2)$$

where w_{ij} represents pairwise weights between a center and its neighbors. Equivalently, the above equation can be expressed as follows:

$$\mathbf{x}'_i = \sum_{j \in \mathcal{N}(i)} w_{ij} \cdot \mathbf{x}_i + \sum_{j \in \mathcal{N}(i)} w_{ij} \cdot (\mathbf{x}_j - \mathbf{x}_i). \quad (3.3)$$

Now the original convolution is disentangled into two components: $\sum_{j \in \mathcal{N}(i)} w_{ij} \cdot \mathbf{x}_i$ and $\sum_{j \in \mathcal{N}(i)} w_{ij} \cdot (\mathbf{x}_j - \mathbf{x}_i)$. As can be seen, the second term is a generalized version of the discrete Laplacian presented in Section 3.2. By setting $\sum_{j \in \mathcal{N}(i)} w_{ij} = 1$, we obtain:

$$\mathbf{x}'_i = \mathbf{x}_i + \sum_{j \in \mathcal{N}(i)} w_{ij} \cdot (\mathbf{x}_j - \mathbf{x}_i), \quad (3.4)$$

where the second term on the right side of the equation is the discrete Laplacian if we further set $w_{ij} = \frac{1}{|\mathcal{N}(i)|}$.

3.3.2 Examples of w_{ij}

Apparently, the most important parameter in Equation 3.4 is w_{ij} . We are interested in what various w_{ij} can encode in the context of Equation 3.4 because it may give us a clear picture of the behavior of various types of discrete Laplacians. To this end, we analyze what in effect the Laplacian encodes by visualizing filtering input, output, and the Laplacian separately. Specifically, we use an image which is a digital surface model generated from a point cloud to conduct the experiment because images are easier to understand compared with 3D point clouds. Three types of filters are demonstrated: average, bilateral, and max filters. The results are shown in Figure 3.4. As can be seen, the

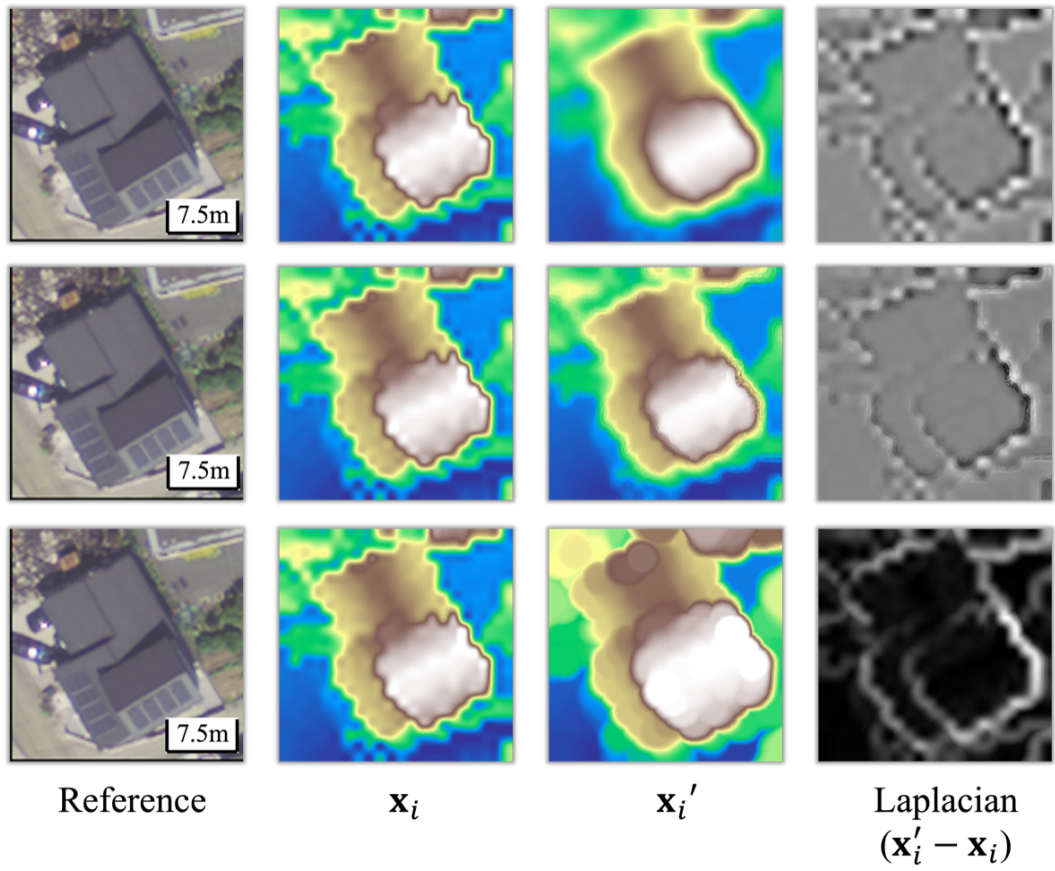


FIGURE 3.4: Visualizations of Laplacians from various filtering results. From the top to bottom rows: average, bilateral, and max filters.

average filter simply performs smoothing to the original input image whereas the bilateral filter smooths the image while retaining details. As for the max filter, it smooths out while dilating pixels. A key observation is that, despite their different effects, the Laplacian component strongly responds to where pixel values change rapidly, i.e., edges. Therefore, the Laplacian component can be regarded as performing edge detection. Therefore, selecting different w_{ij} amounts to selecting different edges.

3.3.3 Formulation of Laplacian Unit

As shown in the previous section, setting different w_{ij} results in different edge detection results based on the discrete Laplacian. For our purpose, the desired behavior is that the selection of w_{ij} can be performed in a fully automated and self-adaptive manner such that the desired edges are detected. To this end, now we formally define LU as below:

$$\mathbf{x}'_i = \mathbf{x}_i + \mathcal{F}(\Delta \mathbf{x}_i). \quad (3.5)$$

The overview of the computation of LU is shown in Figure 3.5. On

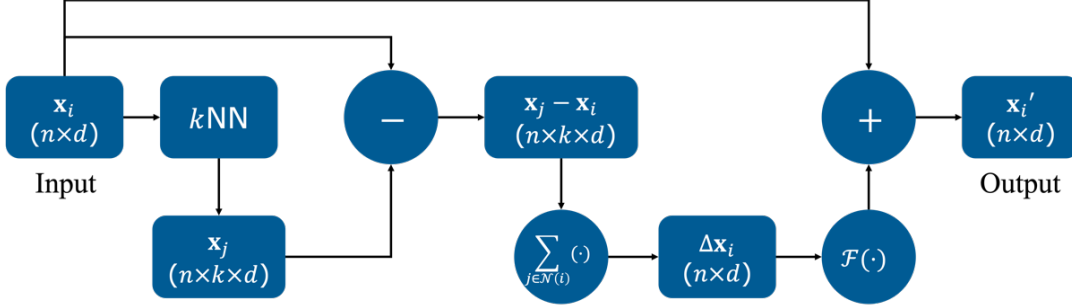


FIGURE 3.5: The computation of LU.

Method	Equation
Convolution	$\mathbf{x}'_i = \sum_{i \in \mathcal{N}(i)} w_{ij} \cdot \mathbf{x}_j$
Disentangled convolution	$\mathbf{x}'_i = \sum_{j \in \mathcal{N}(i)} w_{ij} \cdot \mathbf{x}_i + \sum_{j \in \mathcal{N}(i)} w_{ij} \cdot (\mathbf{x}_j - \mathbf{x}_i)$
Laplacian Unit	$\mathbf{x}'_i = \mathbf{x}_i + \mathcal{F}\left(\frac{1}{ \mathcal{N}(i) } \sum_{j \in \mathcal{N}(i)} (\mathbf{x}_j - \mathbf{x}_i)\right).$

TABLE 3.1: Comparison of convolution, disentangled convolution, and Laplacian Unit.

constructing LU, first, we introduce a nonlinear learnable selectivity function

$\mathcal{F} : \mathbb{R}^d \rightarrow \mathbb{R}^d$ with which we transform the raw discrete Laplacian. The comparison between the above equations is shown in Table 3.1. Since the raw Laplacian encodes the smoothness of the local region, the selectivity function can explicitly control whether to magnify the smoothness (or to make it more discontinuous) or make it smoother. Consequently, the selectivity function learns to enhance the useful discontinuity feature so that this discontinuity has a greater effect on the feature representation of the center point i . Hence, we expect that the useful discontinuities can make the representation of the point i more discriminative. The useful discontinuities in our case may be the deformations on the roof or the boundary of the building that can simply indicate the boundary between the building and the surroundings or inclinations of the building. On the other hand, other discontinuities like tree crowns should be smoothed since it does not convey useful information for CBD. In practice, we implement $\mathcal{F}$ with sequential applications of a linear transformation, a Batch Normalization (BN) (Ioffe and Szegedy, 2015) layer, and a rectified linear unit (ReLU) (Glorot, Bordes, and Bengio, 2011). The linear transformation provides a basic capability to enhance Δx_i selectively. BN and ReLU are applied mainly to facilitate the training by regularization and sparsity, following the common practice of modern DNN designs. The conceptual effect of the selective function $\mathcal{F}$ is shown in Figure 3.6.

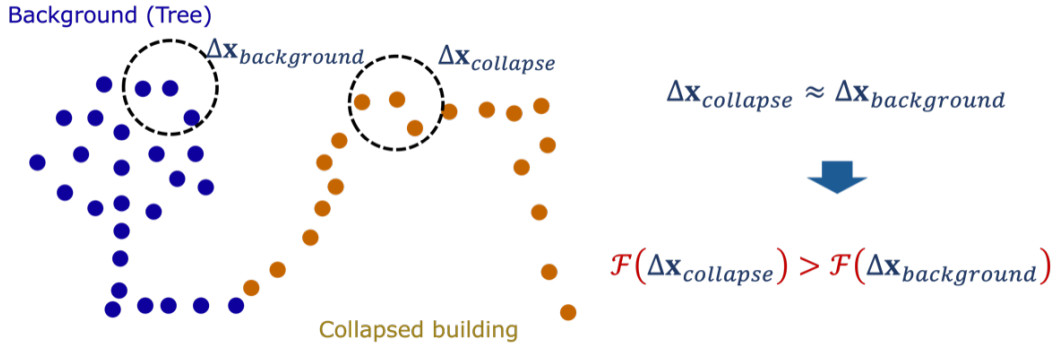


FIGURE 3.6: The concept of the selectivity function $\mathcal{F}$.

Second, the inherited x_i term in Equation (3.5) can be regarded as a residual connection in terms of deep learning. It is well-known that the residual connection can make optimization easier because the backpropagated gradient can go through the residual connection to reach the previous layer without being modified (He et al., 2016a). As a result, the gradient does not vanish and the network architecture can be optimized effectively. In a similar spirit, we use the residual connection to ease the optimization. In addition, the residual connection also ensures that information like global contexts does not vanish. The

discrete Laplacian is a feature that describes local properties around the center point i . Therefore, the property computed by the discrete Laplacian is always relative to the center point ($\mathbf{x}_j$ relative to $\mathbf{x}_i$). However, purely local properties are not sufficient to describe the damage because local properties do not consider global contexts such as scene layout. For instance, both damage-like patterns on a tree and near a building are likely classified as damage without global contexts since they are locally similar. Thus, introducing a residual connection ensures that the output of LU is always a combination of the local and global features. As a result, networks equipped with LUs become aware of local damage patterns without losing global contexts.

Although simple, these modifications are found to be vital in the design of LU. The verification is presented in Section 4.2.2. Furthermore, we verify that the extracted discontinuities by the selectivity function correspond well to damaged parts in Section 4.2.2.

3.3.4 How complicated for a neural network to learn LU?

Since a neural network is a universal approximator, one may expect that the network can implicitly learn LU to improve performance. Also, the Laplacian filter can also frequently be observed from automatically learned filters of CNN. A question is, is LU can automatically learned by neural networks?

Based on Equation (3.2), a simple Laplacian filter can be approximated by setting $w_{ii} = -|\mathcal{N}(i)|$ and other $w_{ij} = 1$. Therefore, it may be relatively easy to construct a Laplacian filter using convolution layers. On the other hand, how to construct LU is not straightforward from Equation (3.2). Equation (3.3) shows that we can disentangle the equation into a form that is very similar to the one of LU (Equation (3.5)). As we can see, to construct LU, there are at least two conditions that need to be met simultaneously:

- $\sum_{j \in \mathcal{N}(i)} w_{ij} = 1$
- The second term in Equation (3.3) must approximate $\mathcal{F} \left(\frac{1}{|\mathcal{N}(i)|} \sum_{j \in \mathcal{N}(i)} (\mathbf{x}_j - \mathbf{x}_i) \right)$ in Equation (3.5).

Obviously, conditions 1 and 2 contradict. Therefore, LU cannot be approximated by a simple convolution layer. Therefore, we believe that LU is not an easy function to be learned by convolution layers. This is verified in Section 4.4 in which the experiment results show that LUs improve performance significantly.

3.4 Laplacian-Enhanced Neural Network

In this section, we elaborate on the design and construction of LE-Net, a network dedicated to collapsed building detection. The overview of the network architecture is shown in Figure 3.8. LE-Net is composed of LUs and commonly used layers in 3D point cloud analysis. In general, we follow the overall framework of PointNet++ (Qi et al., 2017b), the standard encoder-decoder architecture for 3D point cloud analysis, to design LE-Net. The detailed diagram of the architecture of LE-Net is presented in Figure 3.7, which is directly comparable to PointNet++ architecture shown in Figure 1.8. The comparison shows that the major difference between the two models is the introduction of LUs. The encoder is responsible for abstracting the features in a local to global fashion whereas the decoder reconstructs the features from the output of the encoder so that the output of the decoder has the same resolution as the input. Then, the reconstructed features are used to predict the point scores.

We first elaborate on the design of basic blocks that are used as building blocks of LE-Net. Then we stack those blocks and construct LE-Net.

3.4.1 LU-MLP and downsampling block

As demonstrated by the tremendous success of CNN in image processing, local feature learning is indispensable for a successful DNN. Here we construct a feature learner that performs local feature learning while enhancing the important feature representations by LU. Our local feature learner is termed LU-MLP block, which consists of a Group MLP operation, a max-pooling operation, and an LU. The Group MLP applies shared MLP in local groups centered by corresponding center/query points and the max-pooling aggregates the transformed features onto the query points. The transformed points are then further processed by LU to enhance useful features like building damage. Let $\mathbf{p}_i$, $\mathbf{u}_i$ be the 3D coordinates and the feature of point i , respectively. The above process is formally defined as follows:

$$\mathbf{x}'_i = \text{LU} \left(\max_j \left(\phi \left(\mathbf{p}_j \parallel \mathbf{u}_j \right) \right) \right), \quad (3.6)$$

where $\parallel$ represents the feature concatenation and $\phi : \mathbb{R}^{d+3} \rightarrow \mathbb{R}^d$ is an MLP. $\mathbf{p}_j$ and $\mathbf{u}_j$ are the 3D coordinates and features of the neighbor point j . The concatenation of $\mathbf{p}_j$ and $\mathbf{u}_j$ ensures that the output of ϕ is local geometry-aware, which is a widely adopted practice (Qi et al., 2017b; Liu et al., 2019). The role of ϕ is to make point features sufficiently discriminative in preparation for LU.

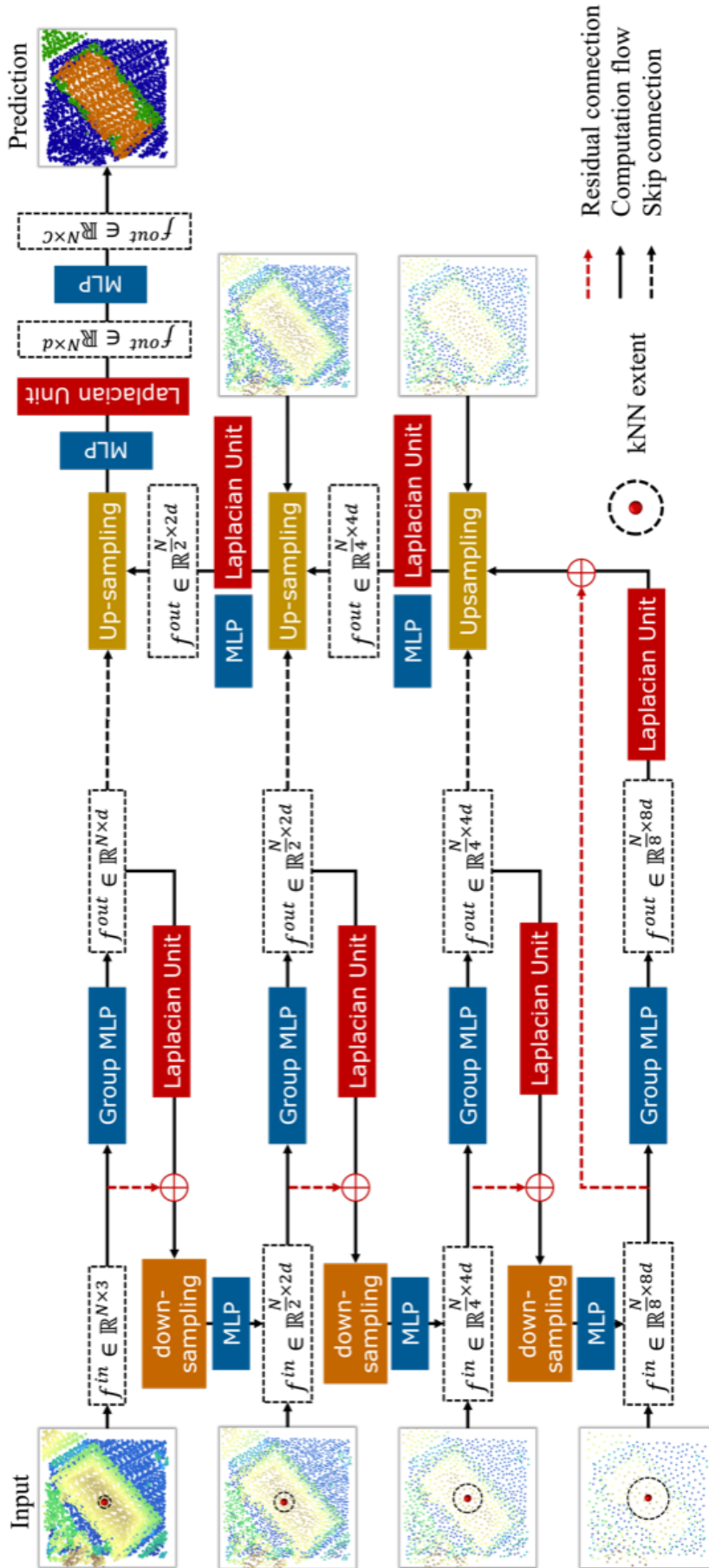


FIGURE 3.7: Detailed architectural of LE-Net. The major difference between PointNet++ and LE-Net is the introduction of LUs and residual connections.

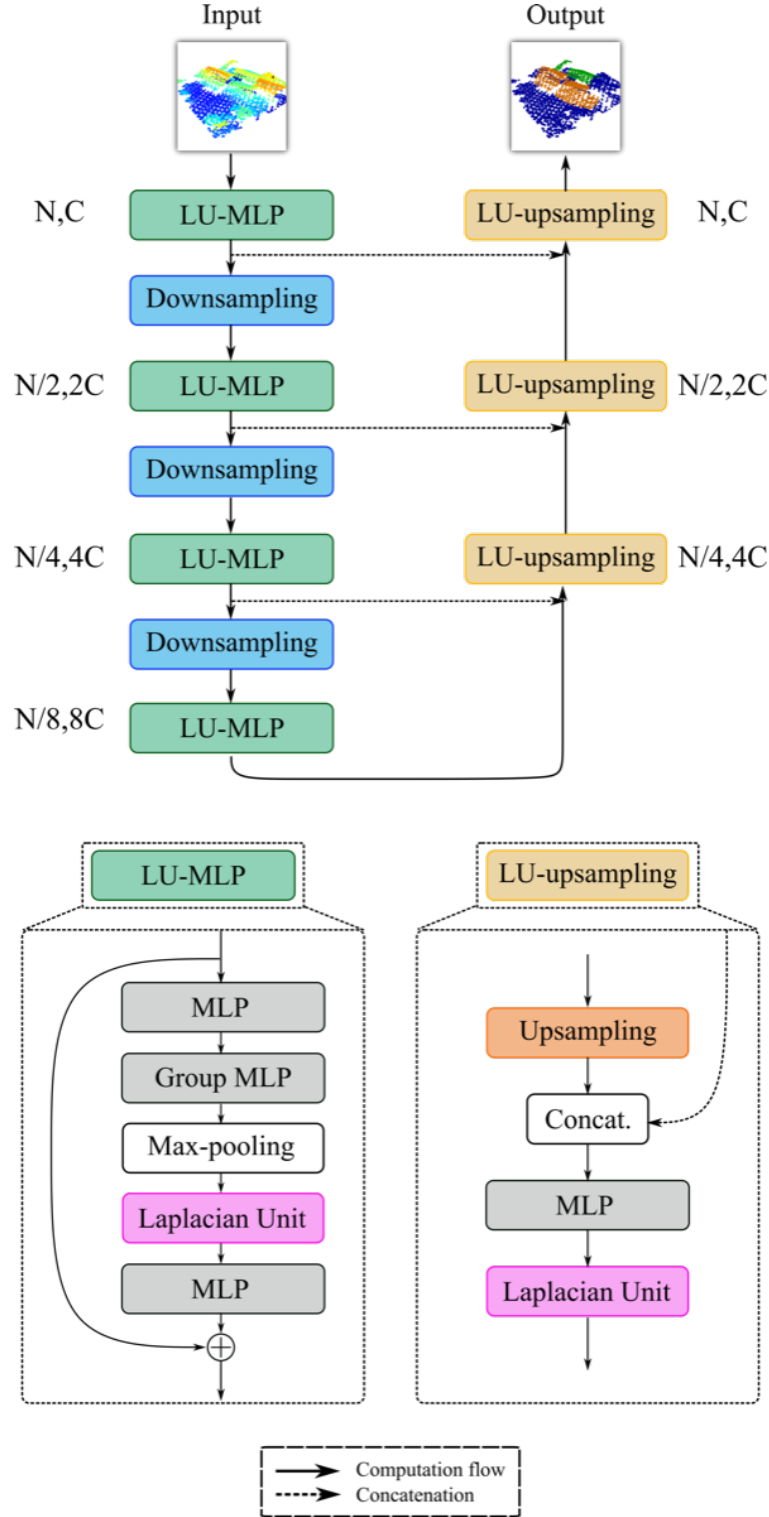


FIGURE 3.8: Network architecture of LE-Net. Left: the concrete design of LE-Net. Right: structures of LU-MLP block and LU-upsampling block. N and C denote the number of input points and the dimension of feature channels, respectively. C is set to 128 by default. Concat. denotes the feature concatenation. The first and the last MLPs in the LU-MLP block perform dimension reduction and expansion, respectively, following the design of the bottleneck residual block (He et al., 2016a). The input point cloud is color-coded by its height for visualization.

Subsequently, the features are enhanced by LU. To reduce the computation cost without compromising performance, we wrap the Equation 3.6 using the bottleneck residual block (He et al., 2016a). Specifically, an MLP is applied at the beginning and the end of Equation 3.6. These two MLPs perform dimension reduction and expansion, respectively. Lastly, the feature after the expansion is added to the original input. The architecture of the LU-MLP block is shown in Figure 3.8.

To achieve hierarchical learning, the input point cloud needs to be down-sampled along the hierarchy while increasing the receptive field (k of kNN) after each downsampling. This process is illustrated in Figure 3.9. As can be seen from the figure, the point cloud gets sparser gradually in order to achieve efficient hierarchical learning. Meanwhile, the kNN radius grows so that extracted local patterns get abstracted along the hierarchy, leading to features that are aware of both local and global features. In practice, we use the farthest point sampling for downsampling.

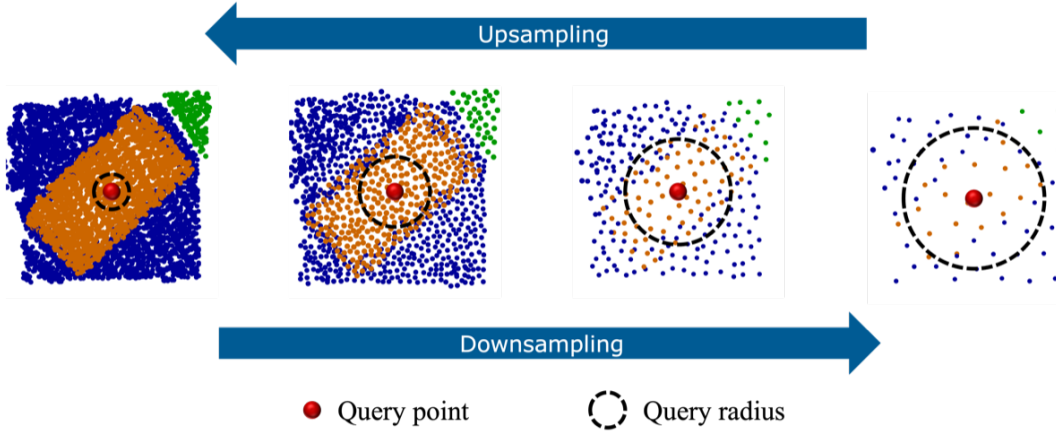


FIGURE 3.9: Illustration of the successive sampling process and the change of receptive fields.

3.4.2 LU-upsampling block

Since we formulate CBD as semantic segmentation, the point resolution must be recovered after the abstraction by the encoder. To this end, we design the LU-upsampling block that recovers the resolution while making feature representation enhanced at the same time. Concretely, the input point is upsampled using three-neighbor interpolation. Then, similar to U-Net (Ronneberger, Fischer, and Brox, 2015), the resulting interpolated feature is concatenated with the features of the same resolution level from the encoder using a skip connection to assist feature reconstruction. The concatenated feature is further

transformed by an MLP and LU so that the useful features remain enhanced after reconstruction.

3.4.3 Constructing LE-Net

The encoder consists of sequential applications of LU-MLP and downsampling blocks. In each resolution level, an LU-MLP is applied to the input followed by a downsampling block. The process is repeated until we obtain a sufficiently discriminative global representation of the input point cloud. In practice, we apply the process three times (see Figure 3.8). For the decoder, we apply three LU-upsampling blocks to fully recover the original resolution that is previously reduced by the encoder for hierarchical learning.

The input channel width (denoted C in Figure 3.8) is set to 128 by default. The downsampling factor is 2, meaning that the number of points is halved for each downsampling. The channel reduction rate in the bottleneck residual block is set to 2, meaning that dimension reduction MLP halves the channel, and expansion MLP recovers it. The neighborhood size of kNN is set to 12 for both Group MLP and Laplacian Unit as we find that the minimum number of points a building has in the Mashiki dataset after down-samplings is 12.

3.5 Summary

In this chapter, we have performed an analysis of the nature of building damage and have shown how the damage can be reliably represented numerically using the discrete Laplacian. Meanwhile, the analysis of the discrete Laplacian reveals a problem. The problem is the lack of selectivity, making damage indistinguishable from surrounding objects that have a similar pattern as damage. This sensitivity is needed to differentiate the damage from the others. To achieve this, we propose LU that can enhance the feature representation of the damaged part in order to facilitate CBD with increased selectivity. Using LUs as core operators, we propose and construct LE-Net, a network dedicated to CBD. We believe that LE-Net is can be a powerful network that can perform at its excellence in detecting collapsed buildings.

Chapter 4

Collapsed Building Detection Using LE-Net

4.1 Introduction

In this chapter, we perform a comprehensive assessment of LE-Net on the CBD task. The outline is shown in Figure 4.1.

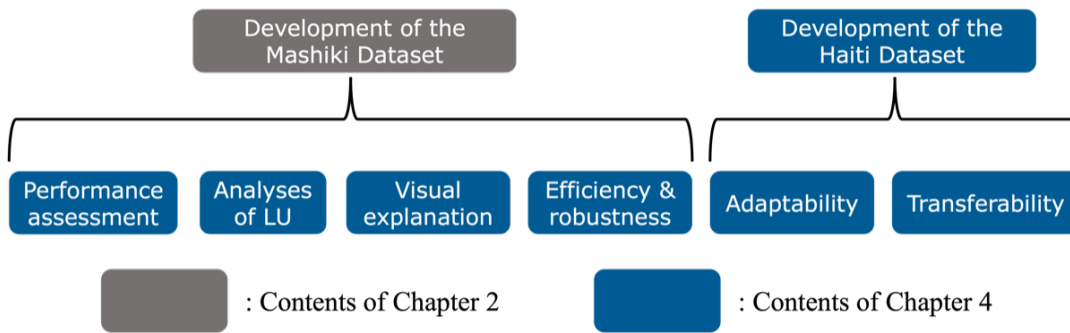


FIGURE 4.1: The outline of Chapter 4.

In the first section, we exclusively use the Mashiki dataset to perform experiments. The quantitative analysis is conducted to evaluate the general performance of LE-Net. Its performance is compared with the same baselines that are used in Chapter 2. The result shows that LE-Net outperforms other methods significantly. Next, a qualitative analysis is performed to investigate the improvements brought by LE-Net with a special focus on the comparison with PointNet++. As a result, LE-Net excels at detecting collapsed buildings that tend to be missed by PointNet++. Furthermore, we perform various analyses on LU, the core operator of LU. Specifically, the usefulness of each component of LU is validated and the effect of the selectivity function $\mathcal{F}$ is investigated because of its great impact on the performance. We additionally perform a visual explanation analysis in order to reveal how LE-Net understands building

damage and how mistakes are made. We consequently understand that LE-Net makes decisions by relying on damage-related parts of the target building. Further, the analysis of its failure modes reveals that mistakes are due to the relatively complete shape of collapsed buildings. Then, we investigate the general aspect of LE-Net including computational efficiency and robustness against the density variation.

In the second part, we investigate the adaptability and transferability of LE-Net. Adaptability refers to LE-Net’s ability to perform well in a substantially different domain. Although LE-Net trained on the Mashiki dataset may work well in a similar region as Mashiki town, the characteristics of the affected region and buildings within may differ significantly from place to place. For instance, the trained LE-Net may fail completely in regions where the major building structures are not wooden (e.g., concrete). Therefore, the adaptability of LE-Net to different events needs to be validated. Transferability, on the other hand, refers to the ability of a DNN to extract transferable knowledge (feature representations) in one domain/task and adapt it to improve performance in another domain/task (Jiang et al., 2022). The transferability of a model in CBD is important because a large number of good-quality training samples are often expensive or prohibitive to make in an emergency scenario. A model with good transferability in CBD means that the knowledge learned in one event is adapted to another event to improve the detection with limited training samples.

To this end, we create a dataset, termed the Haiti dataset, to validate the transferability and adaptability of LE-Net. The Haiti dataset is developed using the data acquired after the 2010 Haiti earthquake. First, the Haiti dataset, a dataset with significantly different characteristics from the Mashiki dataset, is developed for performing experiments. Then, the similarity between the Mashiki and Haiti datasets are analyzed and verified. Next, the adaptability of LE-Net is demonstrated by training it using building samples from the Haiti dataset only. Lastly, the transferability of LE-Net is shown by adapting pre-trained LE-Net to the Haiti dataset.

4.2 Performance and Design Analysis of LE-Net

4.2.1 Performance assessment

TABLE 4.1: Comparison of overall performance (%). The bold texts indicate the best performance.

Method	OA	mPrecision	mRecall	mIoU
PointNet	90.6	80.5	84.0	71.5
PointNet++	92.3	81.4	83.7	72.8
DGCNN	90.3	82.6	82.8	72.0
LE-Net (Ours)	93.4	84.8	85.4	76.0

Quantitative analysis

First, we compare the overall performance of LE-Net with several mainstream deep learning methods for 3D point clouds that are benchmarked in Chapter 2. The result is listed in Table 4.1. As listed, LE-Net achieves the best performance in terms of all performance metrics. Specifically, LE-Net achieves 1.1% higher OA than the second-best score (PointNet++), meaning that LE-Net’s prediction is accurate in general. In terms of average precision, LE-Net significantly outperforms DGCNN, the second-best network, by 2.2%. Therefore, among predicted samples, LE-Net is the most network that predicts most of the samples well. Further, LE-Net achieves 1.4% higher mRecall than PointNet, which means LE-Net has successfully found more building points among the actual points. Since LE-Net is superior in terms of the above three metrics, it naturally achieves the best mIoU, which reflects the overall quality of the predictions. Concretely, LE-Net achieves 3.2% better IoU than the second-best PointNet++, achieving the best score by significantly outperforming it.

TABLE 4.2: Detailed comparison of LE-Net (Ours) and PointNet++. Texts in parentheses indicate improvements made by LE-Net.

Method	Metrics	Collapsed	Non-Collapsed	Background
PointNet++	Precision	58.6	89.8	96.0
	Recall	64.8	92.9	93.3
	IoU	44.4	84.1	89.8
LE-Net (Ours)	Precision	66.9 (+8.3)	91.2 (+1.4)	96.3 (+0.3)
	Recall	67.8 (+3.0)	93.9 (+1.0)	94.4 (+1.1)
	IoU	50.8 (+6.4)	86.1 (+2.0)	91.1 (+1.3)

To investigate the performance of LE-Net more in detail, we compare LE-Net with PointNet++ on which LE-Net’s design is based. Since the design of LE-Net closely follows PointNet++, the comparison can reflect the impact of LU both quantitatively and qualitatively.

TABLE 4.3: Percentage of “Good” detection of collapsed buildings. The prediction of a collapsed building sample is “Good” when IoU for “Collapsed” is greater than 0.5.

Method	Good detection (%)
PointNet++	67.5
LE-Net	78.3

The quantitative comparisons are reported in Table 4.2. As can be observed, LE-Net outperforms PointNet++ in terms of all metrics for all classes. In particular, the improvements are most conspicuous in detecting collapsed buildings. Therefore, LU is the most influential and effective in detecting collapsed buildings.

Concretely, LE-Net achieves 8.3% higher precision than PointNet++, meaning that LE-Net makes significantly fewer false alarms compared to PointNet++. Considering the real-world situation where investigators need to actually design the rescue plan, suppressing false alarms is important because it can prevent the investigators from making redundant plans, which is helpful to achieving efficient rescue. We believe that the suppression of false alarms is the direct consequence of the selectivity brought by LU. The introduction of LU makes it possible to adaptively suppress less meaningful discontinuities. For instance, the tree near a collapsed building may be predicted as “Collapsed” without LU. Therefore, LU’s selectivity allows LE-Net to better discriminate between similar collapsed buildings and similar objects, leading to better precision.

Furthermore, a 3.0% increment in recall compared to PointNet++ is observed. This means that LE-Net is better at finding the actual points belonging to the “Collapsed” class. The increase in recall is valuable because it means that omission is suppressed compared to PointNet++. In reality, omission errors may directly lead to an increased casualty; hence, improving recall is vital in terms of real applications. We believe that this increased sensitivity to the “collapsed” class is also due to the sensitivity of LU. LU adaptively enhances the damaged part so that its feature representation becomes more salient and discriminative.

Since both precision and recall of the “Collapsed” class of LE-Net is greater than those of PointNet++, LE-Net naturally outperforms PointNet++ in terms of IoU as well. In general, high IoU means that the overlapping area of the prediction and the ground truth is large. Therefore, the LE-Net can detect buildings more accurately compared to PointNet++. This is also demonstrated

in Table 4.3 where we report the percentage of “Good” samples. Concretely, we calculate the instance-wise score of collapsed building class, and classified the building that has IoU of “Collapsed” class greater than 0.5 as “Good”. As a result, we find that LE-Net outperforms PointNet++ by 10.8%. Therefore, LE-Net’s quality of CBD is significantly better than that of PointNet++, which in turn validates our motivations.

Interestingly, as for non-collapsed buildings and background classes, we observe that LE-Net consistently outperforms PointNet++. Therefore, the effect of LU is not limited to better detection of damage, that is, LU has general applicability. We conjecture that enhancement/suppression of certain discontinuities brings general improvements. For example, boundaries between non-collapsed buildings and backgrounds are also important for improving performance and is also can be handled by LU. Besides, the ability to suppress discontinuities like noise is certainly useful in general scenarios. However, the improvements in these two classes are not as significant as in the collapsed building class. Therefore, the impact of enhancing/suppressing discontinuities is most useful in improving CBD.

Qualitative analysis

To gain a deeper understanding of how LE-Net enhances the detection, the qualitative analysis is performed and the visualization results are shown in Figure 4.2.

As can be observed from the first row, LE-Net shows a better ability to discriminate buildings and other similar objects. In this case, LE-Net successfully separates the target building and surrounding trees, which leads to improved detection. In contrast, PointNet++ incorrectly recognizes the trees as part of the building. Consequently, the nearby non-residential buildings are incorrectly recognized as collapsed buildings. We believe that the selectivity of LE-Net successfully identifies the tree and suppresses it so that LE-Net can easily identify the boundary between the tree and the building. In practice, this type of error can cause problems when some kind of estimation such as an estimation of debris volume is needed to devise a post-recovery plan.

In the second case, LE-Net is found to be more boundary-aware compared with PointNet++. As can be seen from the second row, the roof of the central building nearly touches the roof of the other building, which makes boundary identification difficult. In this case, PointNet++ produces misclassifications near the boundary. In contrast, LE-Net produces sharp boundary that is faithful to the ground truth, showing its capability to precisely locate and identify

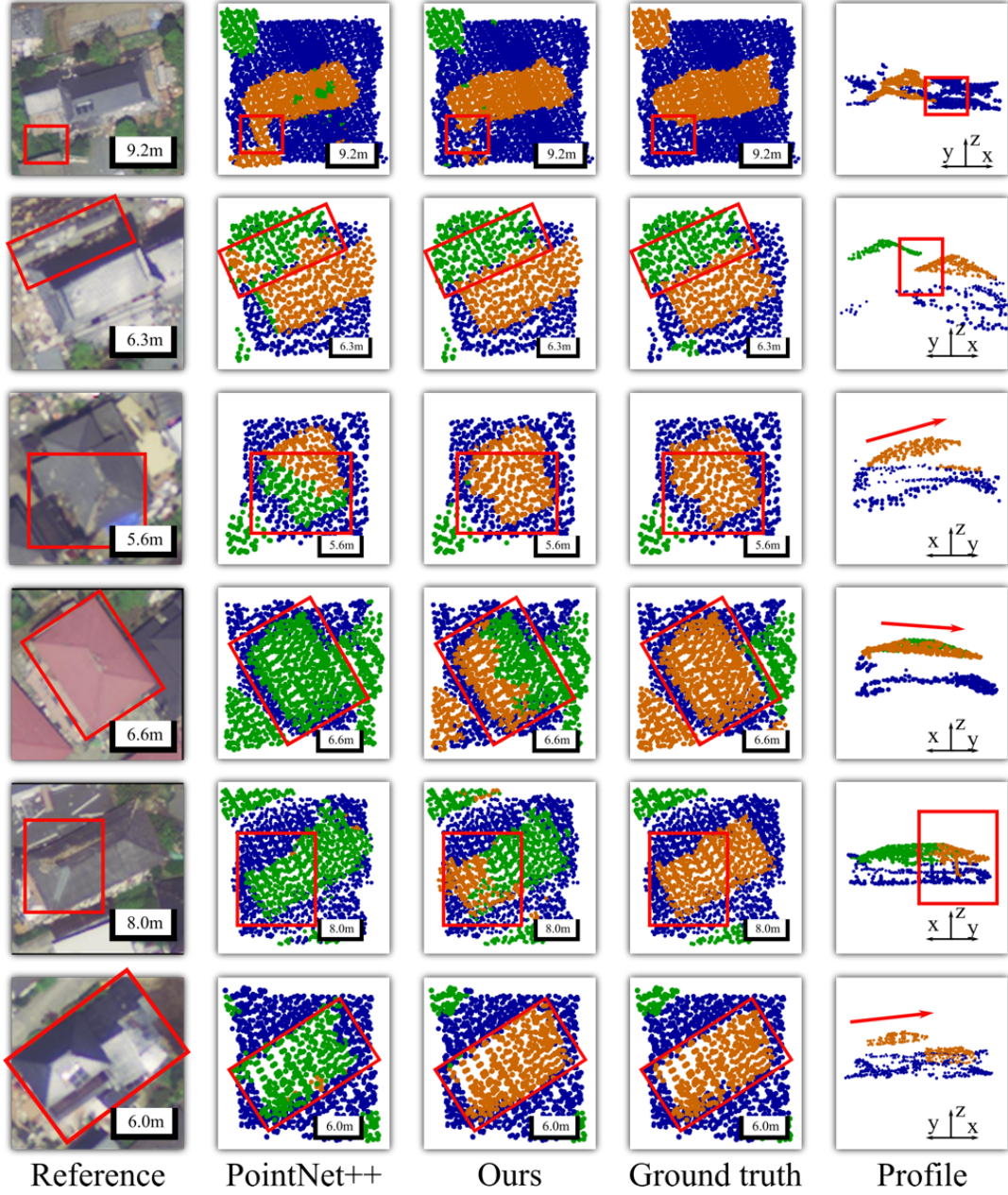


FIGURE 4.2: Qualitative results collapsed building detection on the Mashiki dataset. Green, brown, and blue points indicate non-collapsed, collapsed, and background points. Note that the areas marked by red boxes indicate the improvements made by LE-Net. The “Profile” column shows the horizontal view of the ground truth. The red arrows indicate the inclinations.

boundaries. This point can also be validated by Table 4.2, where the improvement made by LE-Net is also observed in detecting the “Non-collapsed” class.

Further, it is found that LE-Net is less confused with buildings whose roofs are not heavily damaged (the third row). While PointNet++ misclassifies nearly half of the points belonging to the building in the third row, LE-Net successfully produces a smooth prediction. A possible reason is that LE-Net becomes also sensitive in detecting inclinations or debris thanks to the combination of LU and hierarchical learning 3.2.

Moreover, it is found that LE-Net manages to detect buildings that neither undergo severe roof deformations nor inclinations, which are completely missed by PointNet++. For instance, the fourth row shows a case where the building roof is intact while suffering from a slight inclination. In this case, although PointNet++ fails to detect this building, LE-Net manages to correctly recognize the building as collapsed. We think that the use of LU can enhance the debris/inclination patterns so that LE-Net can infer the collapsed building correctly even if its appearance is rather intact.

In the next row, PointNet++ again misses the building entirely whereas LE-Net manages to make a partially correct prediction. As can be observed, the roof suffers from a deformation that is present in the part of the roof. Since the other part of the roof is intact and the inclination is not severe, it may be challenging to identify it as collapsed. We believe that LE-Net successfully detects the deformation and has made correct predictions for the deformed part. Therefore, we conjecture that introduction of LU enhances the sensitivity to small deformations.

In the last case, even if the inclination is not severe, LE-Net manages to produce a near-perfect prediction whereas PointNet++ misclassifies most of the points. Despite the nearly intact appearance, we speculate that LE-Net can detect the small deformations or surrounding debris correctly, which leads to a correct prediction.

4.2.2 Analysis of LU

Component analysis

In this section, we analyze the effect of each component of LU. Specifically, we measure the performance by removing one or multiple components of LU to investigate the impact of the removed components. The result is reported in Table 4.4.

TABLE 4.4: Result of design analysis. mPrecision, mRecall, and mIoU represent class-averaged precision (%), recall (%), and IoU (%). Texts in parentheses indicate the difference in performance between Full (LE-Net) and others.

Method	mPrecision	mRecall	mIoU
Full (LE-Net)	84.8	85.4	76.0
No LU	83.3 (−1.5)	82.9 (−2.5)	73.8 (−2.2)
No $\mathcal{F}$	82.3 (−2.5)	83.6 (−1.8)	73.4 (−2.6)
$\mathcal{F} = \text{Linear}$	83.5 (−1.3)	85.3 (−0.1)	75.1 (−0.9)
$\mathcal{F} = \text{ReLU} \circ \text{BN}$	82.9 (−1.9)	85.4 (+0.0)	74.7 (−1.3)
No residual	83.5 (−1.3)	84.1 (−1.3)	74.4 (−1.6)

First, we validate the effectiveness of LU. As can be seen, removing LU causes significant drops in all performance metrics. Specifically, the drop in precision shows that removing LUs causes more false alarms. The drop in recall shows that removing LUs degrades the LE-Net’s ability to find the correct classes. Since performance drops in all aspects, IoU also reduces significantly. Therefore, LU is indispensable for LE-Net to achieve good performance.

Next, the necessity of the selectivity function $\mathcal{F} = \text{ReLU} \circ \text{BN} \circ \text{Linear}$ in Equation 3.5 is validated. $\mathcal{F}$ is responsible for selectively enhancing useful discontinuities while suppressing less useful ones. Without $\mathcal{F}$, the raw discrete Laplacian is directly fused to the input feature. As we can see from Section 3.2, the raw discrete Laplacian blindly highlights all discontinuities; therefore, it cannot only emphasizes the important one. As expected, removing $\mathcal{F}$ drastically reduces the performance. Somewhat surprisingly, “No $\mathcal{F}$ ” even achieves lower performance compared to “No LU”. It greatly reduces the precision by 2.5%, which is a 1% greater reduction of the precision. In terms of recall, it reduces the performance by 1.8%, 0.7% less reduction compared with “No LU”. We conjecture that the addition of raw discrete Laplacian does increase the sensitivity to those damage since the discrete Laplacian can represent damage; however, the false alarms increase due to the lack of selectivity. Since IoU reduced more significantly than “No LU”, we think that the effect of the raw discrete Laplacian is in general harmful without $\mathcal{F}$.

The picture changes when $\mathcal{F}$ is introduced and it is set to a linear transformation ($\mathcal{F} = \text{Linear}$). We observe a significant increase in recall when $\mathcal{F} = \text{Linear}$. Accordingly, increased selectivity in handling Δx_i , the discrete Laplacian, is vital, especially for finding the actual correct classes. Compared with the “No LU” variant, it improves by about 0.2%. Therefore, a linear transformation is also helpful in reducing false alarms. In general, with the help of

linear transformation, a significant selectivity can be given to LU, which, in turn, enhances the performance greatly. In general, we find that the learnable linear transformation is responsible for most of the performance improvement by LU since the mIoU only reduces by 0.9% when $\mathcal{F}$ is set to a linear transformation.

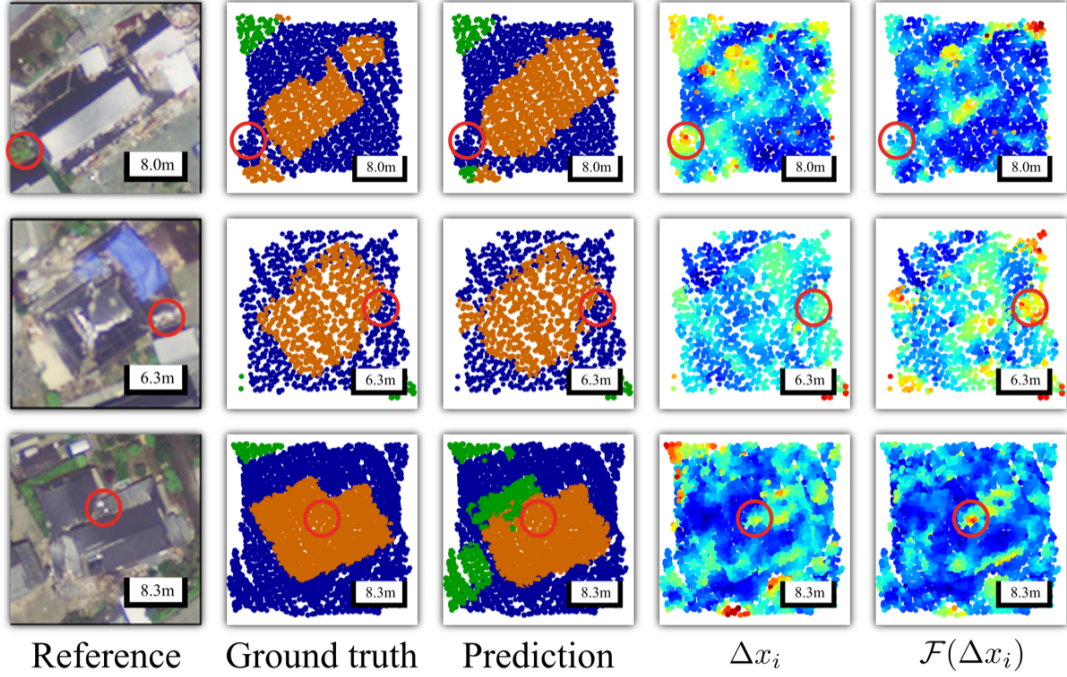
Interestingly, the improvement in optimization ($\mathcal{F} = \text{ReLU} \circ \text{BN}$) also demonstrates a notable improvement in recall. Note that the ReLU function only retains the positive values of features. Therefore, we conjecture that less useful continuities appear in negative values and useful ones in the positive, and the application of ReLU removes those less useful discontinuities while retaining the useful ones, leading to an increased recall. However, the precision reduces more significantly compared to removing LU. Therefore, the lack of selectivity still causes significant false alarms. In general, the improvement from the side of optimization is useful as it improves the performance over the “No $\mathcal{F}$ ” variant. In addition, the best performance is achieved when **Linear** and **ReLU** $\circ$ **BN** are combined, which verifies our design choices.

Finally, we analyze the impact of residual connection in Equation 3.5. As shown in Table 4.4, removing residual connection reduces the scores by around 1.6%. We conjecture that removing the residual connection makes the optimization more difficult (He et al., 2016a). Moreover, recall that LU without the residual connection produces a purely local representation. A local representation without the notion of global contexts may lead to inconsistent predictions. The reduction in performance, however, is not as severe as removing $\mathcal{F}$. We believe that the inconsistency caused by the lack of global contexts is somewhat compensated for by the selectivity provided by $\mathcal{F}$.

Effect of the selectivity function $\mathcal{F}$

We additionally provide visualizations to better understand the effect of the function $\mathcal{F}$ better. Concretely, we visualize the Euclidean norm of discrete Laplacian ($\|\Delta x_i\|$) and the $\mathcal{F}$ -transformed one $\|\mathcal{F}(\Delta x_i)\|$ to investigate the impact of $\mathcal{F}$ on the raw discrete Laplacian. The result is shown in Figure 4.3.

The first row shows that the discrete Laplacian of a tree is significantly suppressed after applying $\mathcal{F}$. Therefore, one of the roles of $\mathcal{F}$, in this case, is to suppress less useful discontinuities like trees since they belong to the same “Background” class and do not need to be enhanced. On the other hand, the discrete Laplacian of the collapsed structure appearing in the right upper corner is significantly magnified. Since the target building is a collapsed building, it is reasonable to enhance this structure so that LE-Net can be more aware of

FIGURE 4.3: Effect of the selectivity function $\mathcal{F}$

the damage. Therefore, $\mathcal{F}$ can perform selective enhancement and suppression simultaneously.

The second row shows a case where the structure of the roof of the target building is relatively complete while touching the ground. In this case, a significant increase of $\|\Delta x_i\|$ is observed in the location where the roof touches the ground and debris. Intuitively, this enhancement is sound because of two reasons: 1) the enhancement leads to a more salient feature representation by combining the global feature (x_i) and local one ($\mathcal{F}(\Delta x_i)$) and 2) the enhancement of the difference makes identification of the fuzzy boundary between the building and the debris easier. On the other hand, we can still observe the enhanced part within the building footprint; therefore, the enhancement can happen within the same class, resulting in enhanced feature representations.

In the third row, we observe a small deformation occur on the roof, which reveals its inner materials. After applying $\mathcal{F}$, the region is enhanced; hence, the feature representation of damage is enhanced. Therefore, LU can accurately locate small deformations by enhanced selectivity brought by $\mathcal{F}$. Meanwhile, we observe that the responses of the location where the roof touches the ground is enhanced while the nearby trees are suppressed. Therefore, we confirm that the role of $\mathcal{F}$ can be very diverse even in the same scene, revealing the effectiveness of learning the selectivity function $\mathcal{F}$.

In order to understand the effect of $\mathcal{F}$ qualitatively, we visualize some examples in Figure 4.4.

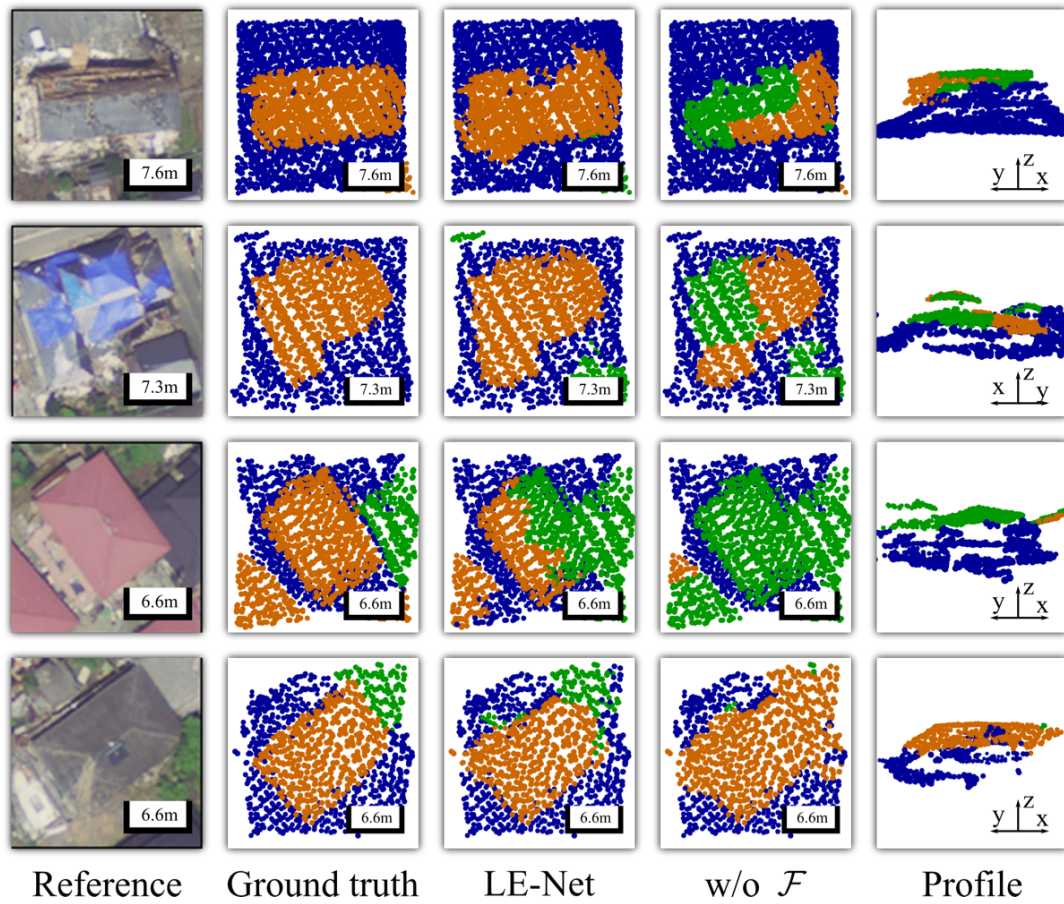


FIGURE 4.4: The qualitative result of LE-Net without the selectivity function $\mathcal{F}$.

The top row shows a case where strong roof deformations occurred on the roof. In this case, we observe that “w/o $\mathcal{F}$ ” fails to detect such deformations. We conjecture that this type of deformation where the inner structure of the roof is exposed is in fact not a very common type of roof deformation in the dataset, which is quite similar to vegetation, and hence LE-Net without $\mathcal{F}$ has difficulty in predicting the destroyed part correctly. In contrast, LE-Net correctly estimates that the deformation part belongs to “Collapsed”. We believe that although the network tends to have difficulty in predicting this part, LU can provide an additional selectivity that can be used to enhance the feature presentation of the damage, which leads to a better prediction.

The second row, on the other hand, shows a case where a building is destroyed because of the ground floor collapse while showing a merely slight inclination. This is a relatively difficult case since the point clouds do not present any strong signs of damage like severe deformations or inclinations. In this case, LE-Net without $\mathcal{F}$ fails to produce the correct prediction for a part of the building. We conjecture that the roof is not heavily damaged and the primary and secondary ones are arranged like non-collapsed buildings, which causes misclassifications. On the contrary, LE-Net successfully produces near-perfect predictions. We think that this is due to the selectivity function $\mathcal{F}$ which can adaptively enhance/suppress the discontinuities. In this case, we think that $\mathcal{F}$ manages to suppress the gap, leading to a more uniform prediction.

The third row shows another challenging situation where the roof is almost intact while presenting only slight inclinations. In this case, the w/o $\mathcal{F}$ variant completely ignores the target collapsed building. Therefore, the raw discrete Laplacian is not sufficient for detecting this type of collapse. In contrast, equipped with $\mathcal{F}$, even in a challenging situation like this, LE-Net successfully predicts 50% of the building, which may be considered a successful detection. We think that there are two ways we can explain this: 1) $\mathcal{F}$ enhances the inclination detection and 2) LU enhances the feature representation of the surrounding damage indicators like debris and collapsed walls. Therefore, we believe that $\mathcal{F}$ is especially useful in detecting slight cases that are easily ignored without it.

The fourth row shows the effectiveness of $\mathcal{F}$ in detecting building boundaries. As can be seen, w/o $\mathcal{F}$ variant fails to separate the target building and neighboring non-collapsed building. This is problematic, for example, when the estimation of debris is needed for devising a post-disaster recovery plan. On the contrary, LE-Net can detect the inter-building boundary well. We believe that the boundary is naturally a discontinuity and $\mathcal{F}$ can enhance this discontinuity

so that the boundary becomes more salient. On the other hand, this finding reveals that the effect of $\mathcal{F}$ is not limited to the “Collapsed” class, demonstrating its generalizability.

4.2.3 Visual explanation

In this section, we investigate how LE-Net understands building damage using visual explanation analysis. Since DNNs work in a black-box manner, explaining the decision of DNNs is essential for applications like CBD in which the reliance of the model on the correct feature (e.g., roof damage) must be guaranteed. We first introduce SEG-GRAD-CAM (Vinogradova, Dibrov, and Myers, 2020), which is used to visualize where segmentation DNNs focus their attention when they make decisions. Then we apply it to LE-Net and perform the analysis.

SEG-GRAD-CAM

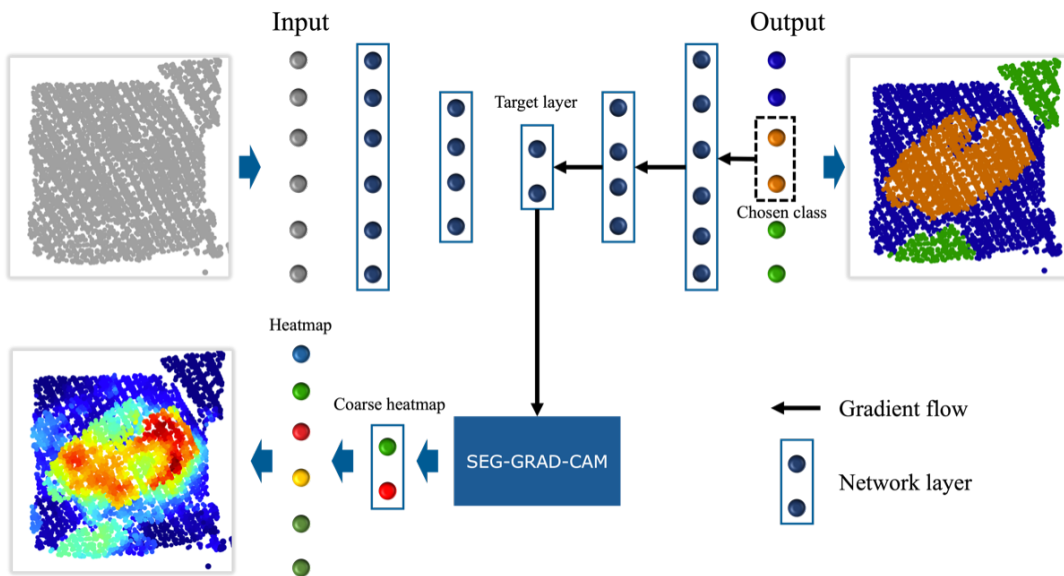


FIGURE 4.5: The concept of SEG-GRAD-CAM (Vinogradova, Dibrov, and Myers, 2020)

In the following, we introduce SEG-GRAD-CAM (Vinogradova, Dibrov, and Myers, 2020), a method that can explain the decision-making of segmentation DNNs. The concept of SEG-GRAD-CAM is illustrated in Figure 4.5.

SEG-GRAD-CAM is an extension of Grad-CAM (Selvaraju et al., 2017), a standard method for visual explanation (Montavon, Samek, and Müller, 2018). Grad-CAM essentially explains the model decisions using a heatmap. The heatmap is calculated based on the backpropagated gradients with an assumption that important features for a particular class possess high partial derivatives

w.r.t the loss of that class. The heatmap is initially generated in the target layer which is defined by the user. The generated initial heatmap is then interpolated to the full resolution for improving visual quality. The hotter regions denote higher importance and vice versa. In essence, it adapts the Grad-CAM to the segmentation setting so that the interpretation of Grad-CAM can be applied to segmentation models. Let $\{A^k\}_{k=1}^K$ be the feature map of the target layer. Let $\sum_{i \in \mathcal{M}} y_i^c$ be the sum of the logits of points belonging to a chosen class c where i indexes the points and $\mathcal{M}$ denotes the set including all points belonging to class c . SEG-GRAD-CAM averages the gradients of $\sum_{i \in \mathcal{M}} y_i^c$ with respect to all N points of each feature map A^k to produce a weight α_c^k to denote its importance. The heatmap can be generated as follows:

$$L^c = \text{ReLU} \left(\sum_k \alpha_c^k A^k \right) \text{ where } \alpha_c^k = \frac{1}{N} \sum_N \frac{\partial \sum_{i \in \mathcal{M}} y_i^c}{\partial A^k}. \quad (4.1)$$

Intuitively, generated heatmaps show the aggregated dependence of points belonging to a chosen class c . For example, hotter regions of the heatmap may highlight roof damage when c is chosen to be “Collapsed”.

Visualizing the decision-making of LE-Net

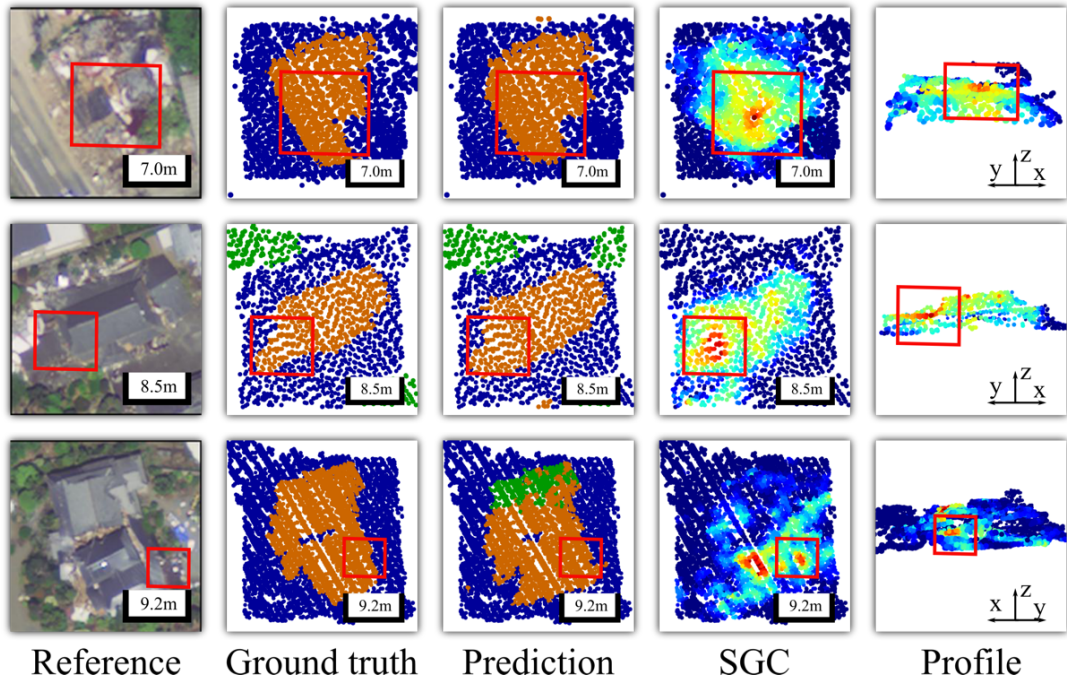


FIGURE 4.6: Visual explanation using SEG-GRAD-CAM (Vino-gradova, Dibrov, and Myers, 2020)

Here we try to visualize the decision-making of LE-Net when it tries to detect collapsed buildings. Hence, the target class is “Collapsed”. We chose the last layer of the encoder as the target layer because the last layer of the encoder is known to represent the most high-level features that are supposed to be very important for the final segmentation result.

The result is shown in Figure 4.6. The figure shows three different cases of heatmaps where LE-Net manages to predict well for all cases. The first row shows a building that undergoes a total collapse where debris appears inside the building’s footprint. In this case, LE-Net puts much attention on the region where the destroyed roof and debris are mixed with each other. Therefore, LE-Net regards the region as the most important for detecting collapsed buildings. This decision is reasonable and intuitive because such a region shows a non-smooth point distribution which represents a typical damage pattern. Notice that the hot region is rather small, which means most of the attention is put on a small region. We believe that LU’s ability to flexibly use the discrete Laplacian enhances the damaged region, which makes the feature representation of that region far more salient than other regions. Furthermore, we found that the model’s attention forms a cluster. The rest of the attention roughly covers the building footprint and extends to its surroundings. Therefore, we can confirm that LE-Net manages to detect the building accurately while making decisions based on its surroundings.

In the second row, the building is collapsed while the structure of the roof remains relatively complete. In such a case, the model puts its attention on the region where the primary and secondary roofs touch each other. A possible explanation is that the discrete Laplacian values on the edge of the primary roof can describe damage well. As analyzed in Section 3.2, the distribution of the strong response of the discrete Laplacian roughly delineates the contour of roofs. In addition, the strength of responses grows with the increasing scale. Considering that the primary and second roofs of non-collapsed buildings are always parallel to each other, we can infer that the feature representations at the edge of the primary roof grow smoothly and stably. On the other hand, the primary and secondary roofs often are not parallel to each other when a building collapses due to structural failure. Thus, the responses of the discrete Laplacian should be irregular. This irregular pattern can also grow with increasing scale; hence, it is a characteristic that is always present. Considering the prevalence of such a pattern in collapsed buildings, we conjecture that LE-Net detects and relies on this particular pattern to make robust predictions especially when roof structures partially or totally withstand the collapse.

In the third row, we again notice that LE-Net highlights the boundary between the primary and secondary roof, which is consistent with the second case. What makes this case different from the above one is that LE-Net also highlights the region where the secondary roof touches the ground. We believe that high dependence on this type of region is sound because we know that usually roofs are always above the ground and we can determine that a building is collapsed whenever the roof touches the ground. Notably, this result is also in line with the damage description of Okada and Takai (2000). We speculate that LU’s ability to selectively enhance the important region helps LE-Net to identify the fuzzy boundary between a rigid roof and non-smooth debris.

Failure modes

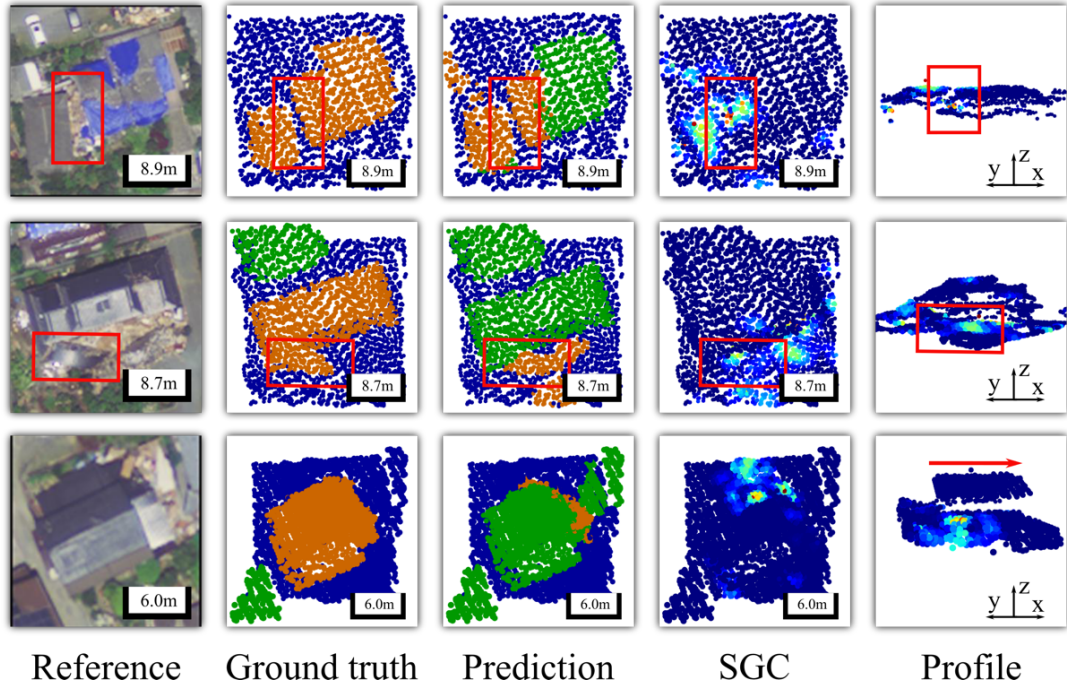


FIGURE 4.7: Failure modes of LE-Net on the Mashiki dataset.

Another useful utility of SEG-GRAD-CAM is to identify failure modes. When the model fails, it is highly likely that its decision is wrong and the heatmap provides some clues about how the model makes an incorrect decision, which may be used to improve the model. Therefore, similar to the above case, we apply SEG-GRAD-CAM to the last layer of the encoder and choose the “Collapsed” class as our target class. We identify several failure models of LE-Net and the result is shown in Figure 4.7.

The first type of failure is that LE-Net sometimes cannot recognize the building as one part when it is broken in the middle. In the first row, LE-Net

can predict the collapsed part well; however, the majority of the other part without visible damage is recognized as non-collapsed buildings. We believe that the separation of the building in the middle makes it difficult for the model to infer the connectivity between the parts. The “SGC” column shows that LE-Net mostly pays attention to the collapsed building and its debris. Therefore, the predictions of the building parts on both sides of the debris are acceptable. On the other hand, we observe that significant misclassifications occur on the major part of the building (right building). This major part of the building appears to be a combination of buildings since we can recognize that the two roofs of them are distinct structures but are closely located. The misclassifications are observed on the right side of this building. We conjecture that LE-Net detects the boundary between two roofs and mistakenly recognizes them as separate buildings. To mitigate this problem, we may use building footprints to postprocess the predictions so that only one category is assigned to a building. Nevertheless, we believe that this type of mistake is not serious since the collapsed part is accurately detected by LE-Net.

In the second case, we observe that the building suffers from a partial collapse. Concretely, the front part of the building collapses and the back of the building is still standing upright. In this case, LE-Net successfully detects the collapsed part as “Collapsed” but it fails to recognize the standing part as the collapsed building. Therefore, LE-Net prioritizes structural completeness over the existence of debris. We believe that this is a logical consequence of excluding debris from the definition of a collapsed building. We believe that this type of mistake, like the one mentioned above, is not fatal because the collapsed part can be detected reliably. Therefore, some postprocessing methods that emphasize the detection of the collapsed part can be used to mitigate this problem.

The third row shows a case where the target building suffers from a pancake collapse. Specifically, a building collapses without significant inclination or serious damage to the roof when it suffers from a pancake collapse. Since such a type of collapse does not provide obvious indicators such as roof deformations or inclinations from which damage can be inferred, the model needs to detect this type of building mainly based on contextual information or very slight damage if it exists. As can be seen, the correct predictions mainly appear where the roof touches the ground, which is reasonable since it is closely connected to debris which is useful contextual information indicating collapse. From the “SGC” column we can also observe that LE-Net successfully locates a slight deformation on the secondary roof. However, since most part of the building

withstands the collapse without significant damage, LE-Net fails to infer the correct damage state. Similar to the second failure case, we conjecture that LE-Net regards structural completeness as more important than the presence of debris. To mitigate this issue, some postprocessing methods that examine the relationship between detected collapse and debris may be implemented.

4.2.4 Computational efficiency

TABLE 4.5: Comparison of computational efficiency. Param. (M) and FLOPs (G) denote the number of parameters and floating point operations per second, with lower scores indicating higher efficiency.

Method	Param.	FLOPs
PointNet	3.5	7.2
PointNet++	3.5	8.6
DGCNN	1.6	5.4
LE-Net	4.3	4.1

In this section, we analyze the computational efficiency of LE-Net and compare it against other methods. Specifically, we report the number of parameters and floating point operations per second (FLOPs). Fewer parameters generally indicate less risk to overfit while fewer FLOPs indicate that the model requires less computation. The result is listed in Table 4.5.

As can be seen, LE-Net uses more parameters compared with others. This is a direct consequence of designing LE-Net based on PointNet++ which has already carried 3.5 M parameters. Although the use of bottleneck residual design in LU-MLP block reduces the computation cost, the addition of LUs at every stage of the network inevitably increases the number of parameters used. However, since we do not observe overfitting during training, we think that the current number of parameters does not affect the practical utility of LE-Net.

On the other hand, we observe that of LE-Net requires the fewest FLOPs compared to the listed methods. Specifically, LE-Net requires around 50% fewer FLOPs compared to PointNet++. We believe that this is due to two facts: 1) LU is by default an efficient operator and 2) the use of bottleneck residual design reduces the FLOPs without compromising performance. Therefore, our design of LE-Net is resource friendly. This trait is valuable in practical application especially when CBD is required for emergency damage mapping where computation resource is very limited.

4.2.5 Robustness to the density variation

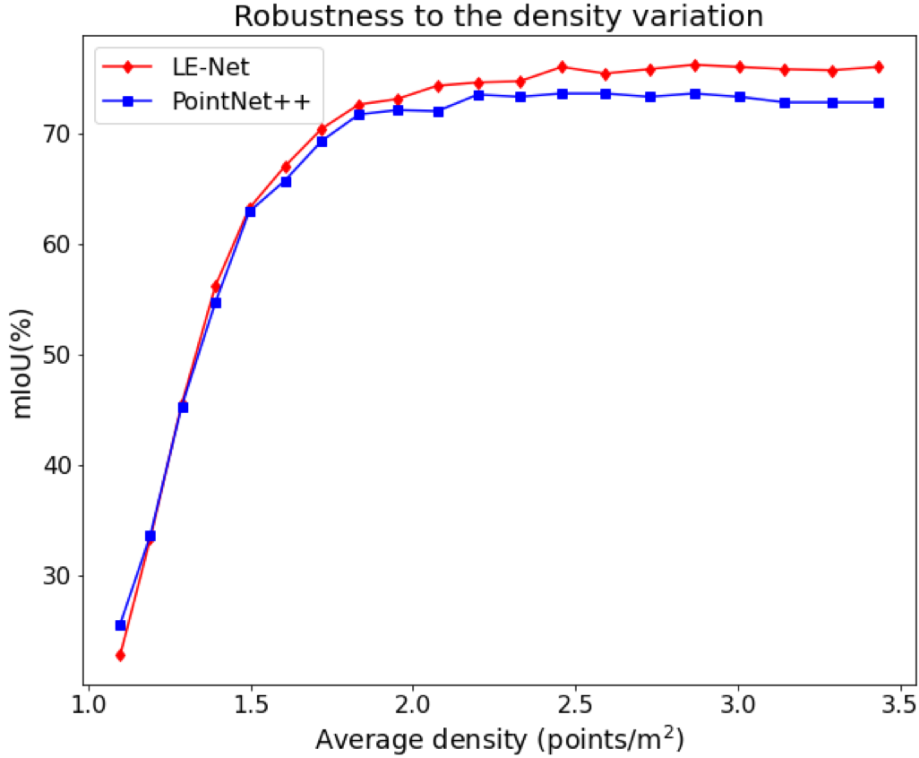


FIGURE 4.8: Robustness to the density variation.

In this section, we analyze the impact of point density. This can on the one hand can quantify the LE-Net’s robustness of LE-Net to density changes and can investigate the generalizability against different sensors on the other. Therefore, the robustness of the model to the density variation is thus important to understand the model’s reusability. To do this, we vary the point density from approximately 1 points/m² to 3.5 points/m² to compare the performance. Note that we do not use any measure that deals specifically with density corruption during the training. The scores of PointNet++ are also reported for reference.

The result is reported in Figure 4.8. We observe from the figure that the LE-Net’s performance drops significantly when the point density is extremely low (around 1 points/m²). Since LE-Net learns from the surface geometry, we believe that making points sparse directly destroys most of the geometry which makes it impossible for the models to perform well. However, we observe that the mIoUs increase rapidly as the point clouds become denser. For instance, when the average density is increased from around 1 points/m² to 2 points/m², the performance increases from 15% mIoU to 74% mIoU. Then, the performance

keeps increasing slowly with increasing density. Therefore, we believe that LE-Net is robust against a certain level of density corruption since 2 points/m² is about 57% of the original density. In addition, the steep increase and subsequent saturation of the performance suggest that there might exist a lower bound that ensures the minimum amount of geometry in order to achieve an acceptable performance. This observation is in fact in line with Critical Point Theory Qi et al. (2017a). In their theory, they state that PointNet recognizes the point clouds by recognizing key points, which are often significantly fewer than the original point clouds. The key points usually form a skeleton of the object of interest, making PointNet robust to a certain level of point corruption/noise. Since LE-Net's design is based on PointNet++, which of course is based on PointNet, we conjecture that such a theory is applicable to LE-Net as well.

We observe that LE-Net performs worse than PointNet++ when the points are extremely sparse. We believe that adversely affects the performance because the major difference between LE-Net and PointNet++ is the use of LU. We conjecture that the lack of geometry makes discrete Laplacian challenging to infer the underlying geometry by discrete approximation. However, with increasing point density, LE-Net quickly recovers the performance. Concretely, LE-Net begins to outperform PointNet++ when the point density reaches around 1.5 points/m². Thus, it is evident that a sufficient amount of geometry is vital in the design of LU.

4.2.6 Effect of neighborhood size k

In this section, we analyze the influence of k , the number of neighbors involved in Group MLP and LU. The maximum number of k is 12 because the minimum number of points after all down-samplings is 12 among buildings in the Mashiki dataset. Therefore, we vary k from 2–12 to compare the performance. As shown in Figure 4.9, we find that $k=12$ is the best choice that brings the best performance. Therefore, we use $k=12$ as the default setting.

4.2.7 Damage mapping under a real-world scenario

In this section, we aim to further investigate the generalizability of trained LE-Net under a real-world scenario. Concretely, we assume that there are no building footprints/locations available to extract buildings and classify point clouds. Therefore, we must apply LE-Net to a "scene" which encompasses the disaster-affected regions and predict all points inside. To achieve this, we first sample some seed points over the entire scene and extract the point clouds by

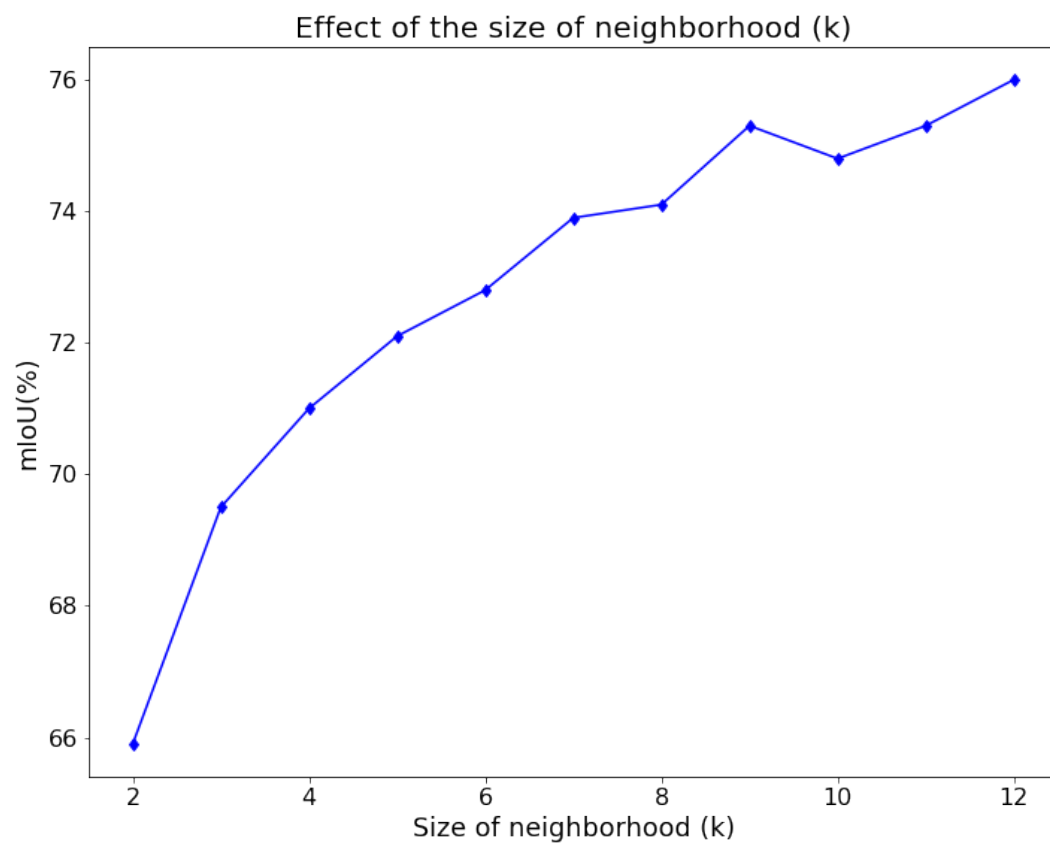


FIGURE 4.9: Effect of neighborhood size k.

treating the seed points as centers. The procedure is illustrated in Figure 4.10. As a result, a number of point clouds that cover the entire scene are extracted and they are subsequently segmented by LE-Net.

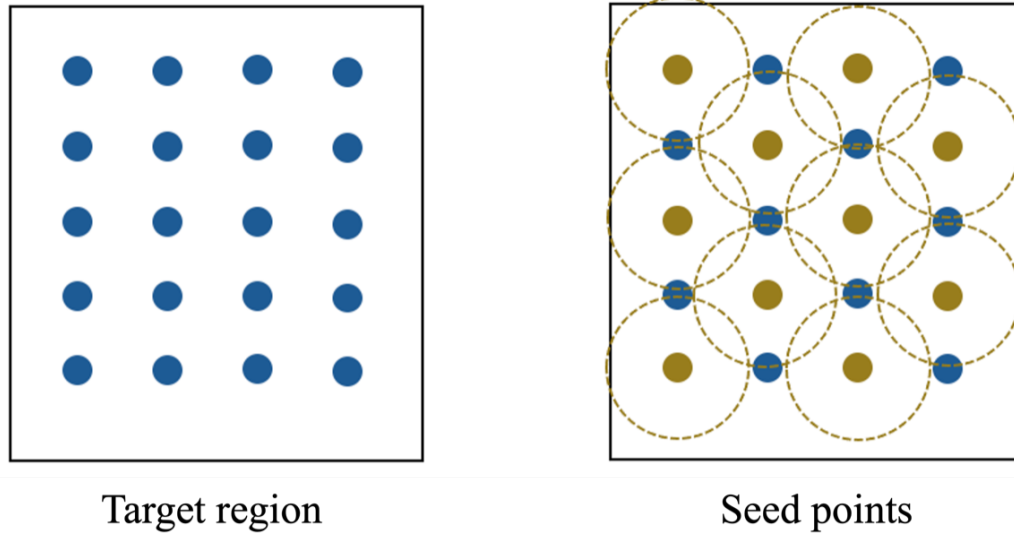


FIGURE 4.10: Illustration of seed point-based extraction.

We choose the Shimada area for this experiment. The overview of this area is shown in Figure 4.11. Shimada is located to the south of the central area of Mashiki-Town. We choose this area for a number of reasons: 1) this area is inside the extent of LiDAR acquisition; 2) there are many buildings collapsed due to the earthquake; 3) a high-resolution Google Earth historical image is available which helps us to identify collapsed buildings.

In selecting seed points, we first perform 3D grid subsampling with a cell size of 8m. For each grid cell, we randomly select a point as the seed point and gather 8,192 nearest points to form a point cloud. Note that we have checked that such a pair of parameters cover the entire region where a point is predicted by LE-Net at least once.

Figure 4.12 shows the overview and details of the prediction results. As can be seen, LE-Net shows an acceptable prediction which agrees well with the Google Earth image in Figure 4.11. Moreover, we confirm that LE-Net can detect collapsed and non-collapsed buildings well. For example, the zoomed-in area contains several non-collapsed and collapsed buildings and LE-Net provides a sound prediction. Therefore, we believe that LE-Net can perform well in regions that are similar to Mashiki Town in real-world situations.

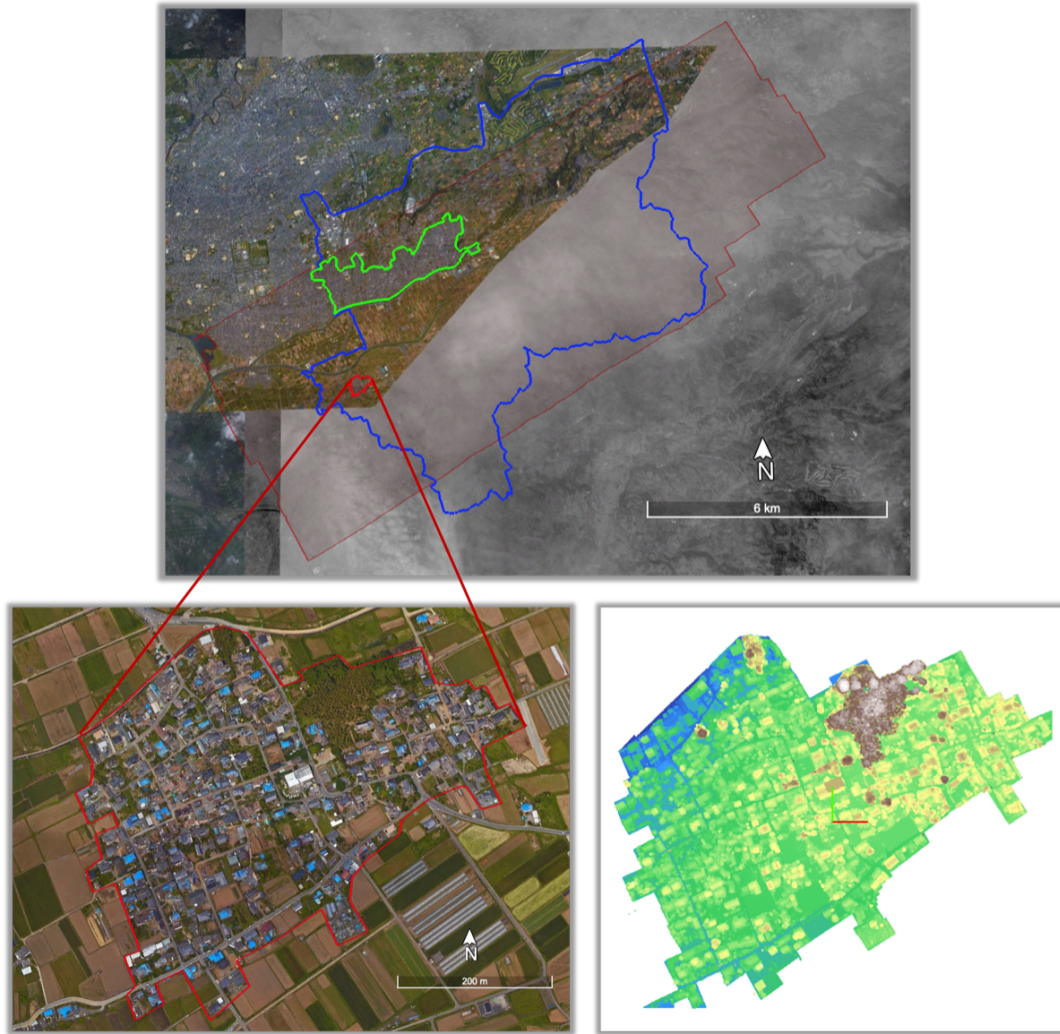


FIGURE 4.11: Overview of the Shimada area (Google Earth). Blue outline: Mashiki-Town. Green outline: the central area of Mashiki-Town. Red outline: the Shimada area. Red shaded area: LiDAR extent. The background is the Google Earth historical image (April 2016). LiDAR point clouds are color-coded by elevation.

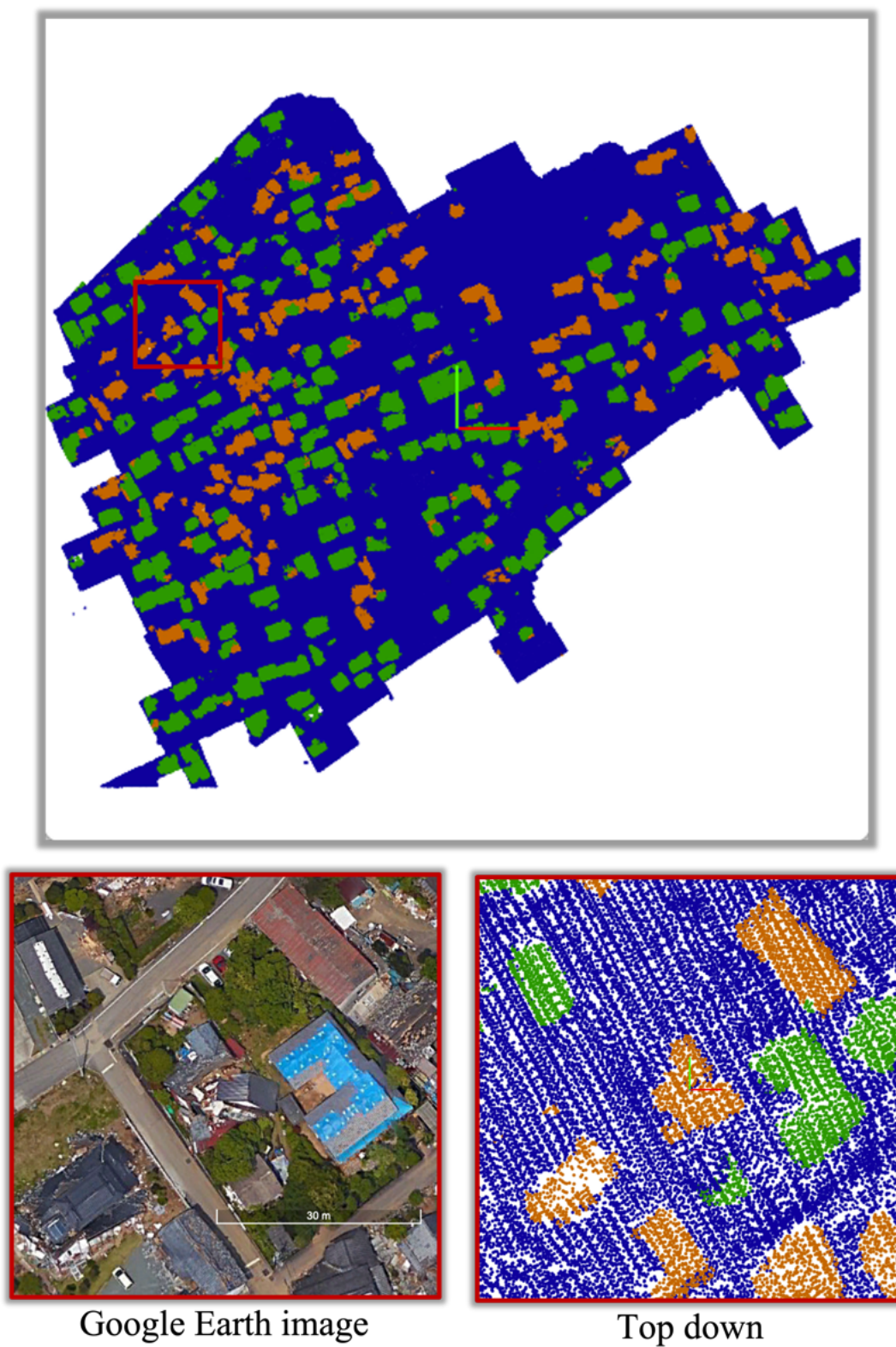


FIGURE 4.12: Result of CBD of Shimada area.

4.2.8 Experiments and discussions on ground deformation induced building damage

In this section, we explore the applicability of LE-Net to ground deformation-induced building damage. Despite that the ground is not a key component of a building, ground deformations such as liquefaction and damage to the retaining wall caused by earthquakes may cause significant damage to buildings. Note that it is often hard to determine whether a collapse is caused by ground deformations or ground shaking since their effects on the building cannot be separated easily. Therefore, we limit ourselves to the exploration of the detectability of building damage in the presence of ground deformations.

Theoretically, LE-Net is trained on the dataset designed based on the damage chart (Okada and Takai, 2000) to separate D5 buildings and others. Therefore, it is reasonable to expect that LE-Net performs well regardless of the existence of ground deformations. In the following, we provide the experimental validation of the above assumption.

For the study area, we choose the area near Mashiki Town Community Hall in which large ground deformations are observed. The overview of the area is shown in Figure 4.13. As can be seen, many buildings in this area are collapsed and infrastructures like roads and retaining walls are severely damaged as well.

In experiments performed in Chapter 2 and 4, this region is included in the training region. Therefore, we change this region to the test region and treat the original test region as the training region. This is illustrated in Figure 4.14.

The results are shown in Figure 4.15. The top row in the figure shows a case where a building suffers from ground runoff. However, the building still stands upright and therefore it is classified as “Non-collapsed”. As can be seen, LE-Net correctly predicts this building as a non-collapsed building. The misclassification occurs at the place where the roof touches the bank. We speculate that this is because the roof of a collapsed building often falls off and touches the ground, causing confusion. For a collapsed building with ground deformations, LE-Net correctly classifies it as “Collapsed”. Misclassification occurs at the place where a part of the road collapses, which is quite similar to a part of a collapsed building. Therefore, we believe that LE-net can operate reasonably well in the presence of ground deformations.



FIGURE 4.13: Overview of the area with large ground deformations. Green: the central area of Mashiki Town. Red: study area where large ground deformations occurred. Gold: Observable ground deformations in this area.

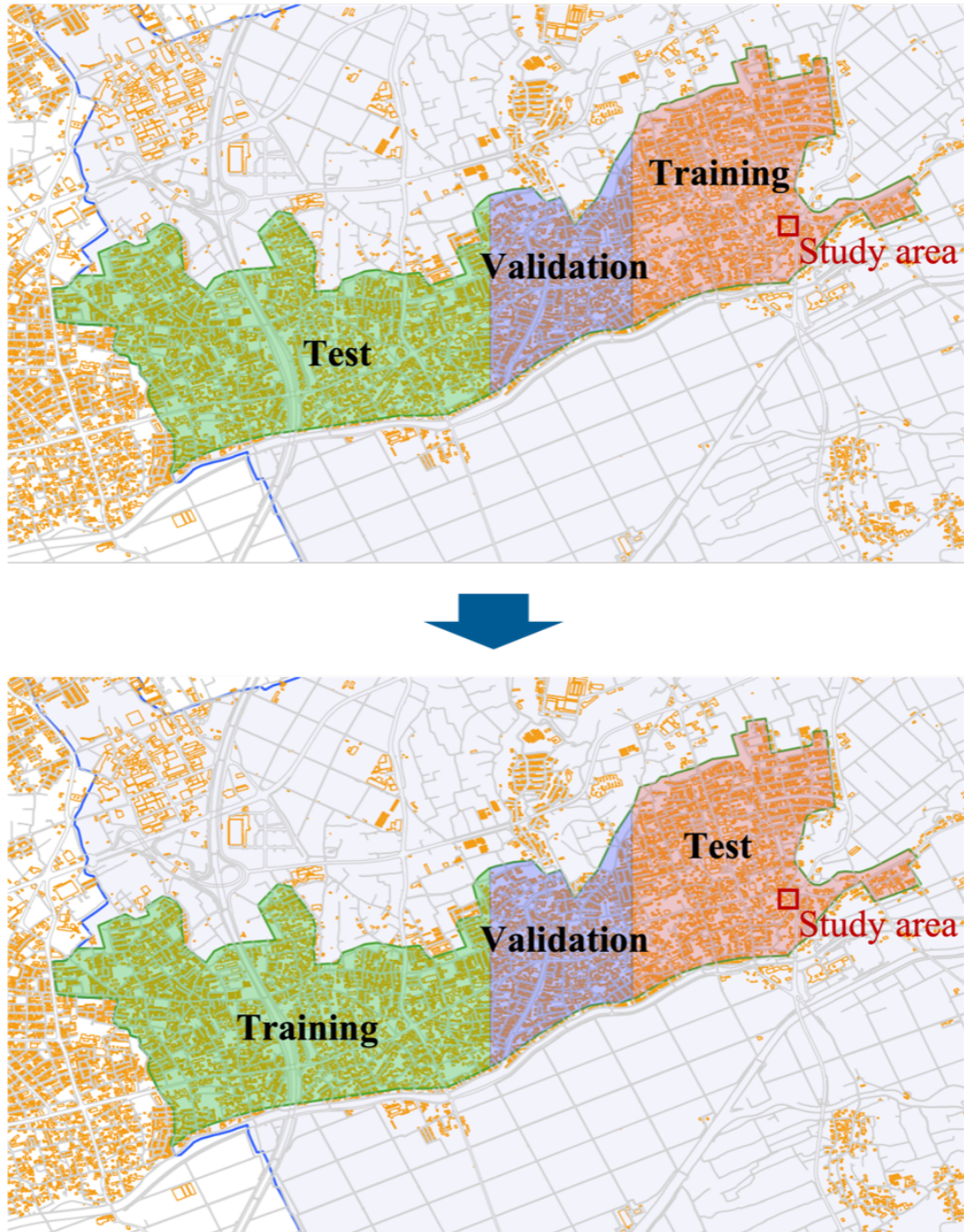


FIGURE 4.14: Change of training and test region. The area with large ground deformations is thereby included in the test region.

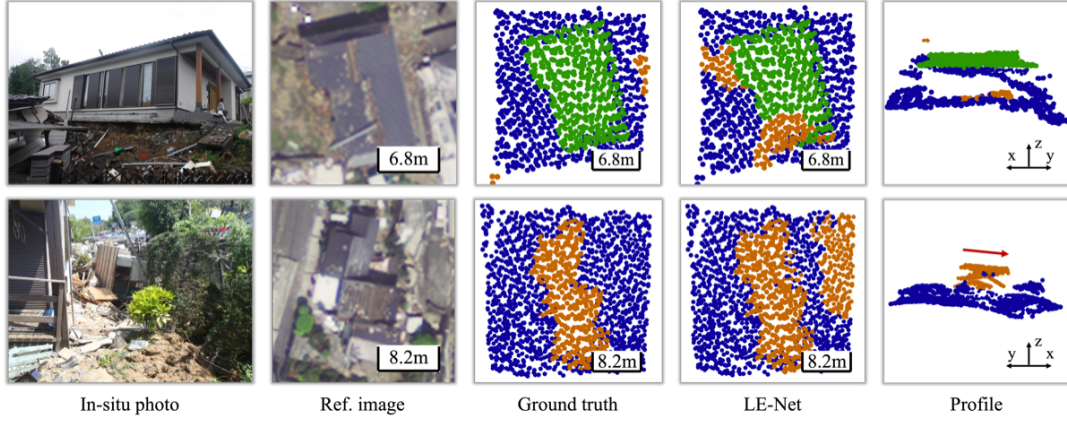


FIGURE 4.15: Result of detection of a collapsed building in the presence of ground deformation. Top: a non-collapsed building ground deformations. Bottom: a collapsed building with ground deformations.

4.3 Adaptability and Transferability of LE-Net

In this section, we validate the adaptability and transferability of LE-Net. The definitions of both terms are described in Section 4.1. This section begins with the development of the Haiti dataset, which is a dataset with substantially different building characteristics. Next, we perform analyses to assess the adaptability and transferability of LE-Net.

4.3.1 Haiti dataset

In this section, data sources and procedures for developing the Haiti dataset are elaborated.

Study area and data sources

The study area is located in Port-au-Prince, Haiti, which was severely struck by the 2010 Haiti earthquake. The earthquake killed approximately 222,560 people and caused significant damage to buildings and infrastructures (World Bank - ImageCat Inc., 2010a). The major structure types of this area are reinforced concrete (RC) and masonry. Therefore, we can validate the generalizability of the model to concrete buildings. As shown in Figure 4.16, the area near Haiti's National palace is chosen for extracting building samples. The area of about 2.7km^2 contains about 2,000 buildings.

After the earthquake, the World Bank, ImageCat Inc., and a network of engineers and scientists performed a damage assessment and produced building footprints by manual photo-interpretation (World Bank - ImageCat Inc.,

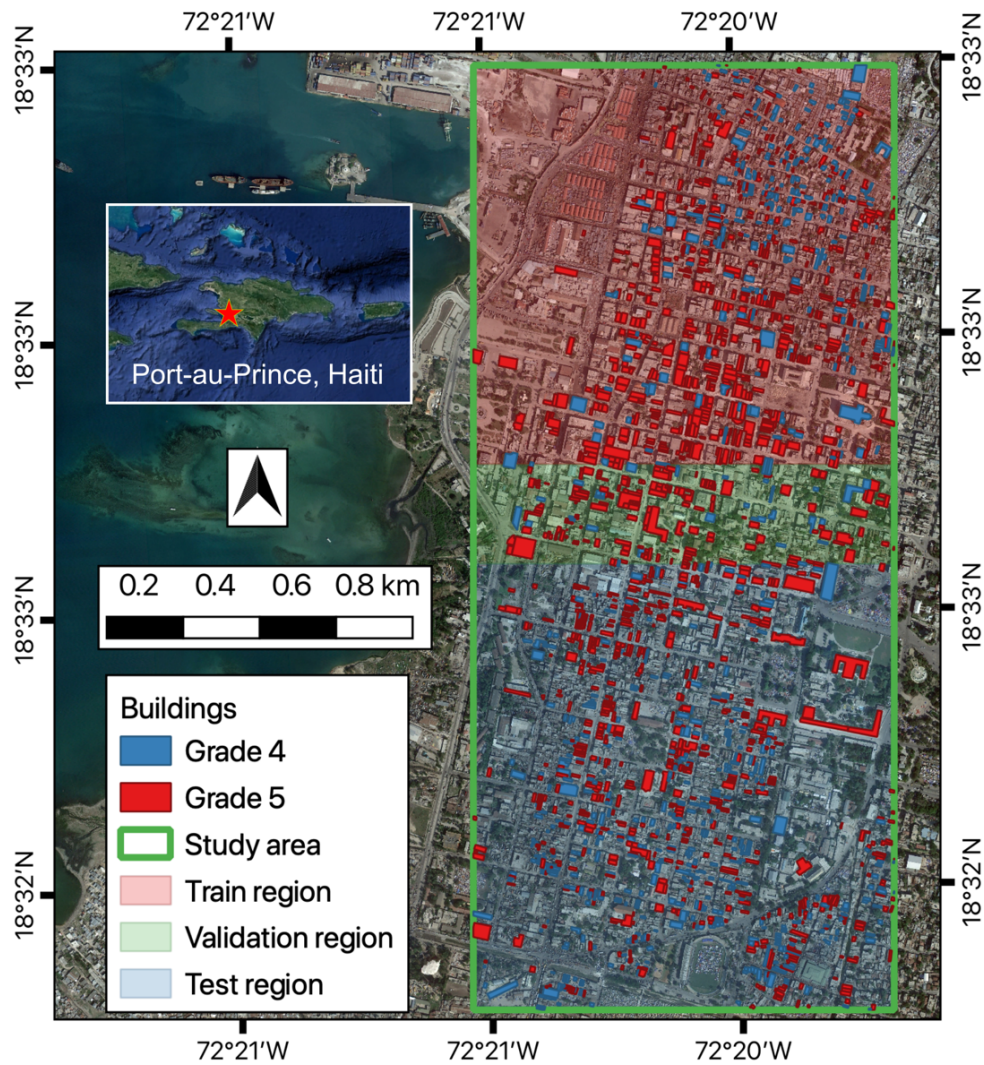


FIGURE 4.16: The study area of the Haiti dataset development. The area in Port-au-Prince near the National Palace of Haiti is chosen for the study area.

2010a). In this assessment, only the footprints of destroyed and very heavily damaged buildings are mapped. The damage interpretation is based on the EMS-98 standard (Grünthal, 1998). The building footprints are shown in Figure 4.16. The very heavily damaged buildings and destroyed buildings correspond to Grades 4 and 5 in EMS-98, respectively. For reference, the classification of damage to buildings according to EMS-98 standard is shown in Figure 4.17.

Classification of damage to masonry buildings		Classification of damage to buildings of reinforced concrete	
	Grade 1: Negligible to slight damage (no structural damage, slight non-structural damage) Hair-line cracks in very few walls. Fall of small pieces of plaster only. Fall of loose stones from upper parts of buildings in very few cases.		Grade 1: Negligible to slight damage (no structural damage, slight non-structural damage) Fine cracks in plaster over frame members or in walls at the base. Fine cracks in partitions and infills.
	Grade 2: Moderate damage (slight structural damage, moderate non-structural damage) Cracks in many walls. Fall of fairly large pieces of plaster. Partial collapse of chimneys.		Grade 2: Moderate damage (slight structural damage, moderate non-structural damage) Cracks in columns and beams of frames and in structural walls. Cracks in partition and infill walls; fall of brittle cladding and plaster. Falling mortar from the joints of wall panels.
	Grade 3: Substantial to heavy damage (moderate structural damage, heavy non-structural damage) Large and extensive cracks in most walls. Roof tiles detach. Chimneys fracture at the roof line; failure of individual non-structural elements (partitions, gable walls).		Grade 3: Substantial to heavy damage (moderate structural damage, heavy non-structural damage) Cracks in columns and beam column joints of frames at the base and at joints of coupled walls. Spalling of concrete cover, buckling of reinforced rods. Large cracks in partition and infill walls, failure of individual infill panels.
	Grade 4: Very heavy damage (heavy structural damage, very heavy non-structural damage) Serious failure of walls; partial structural failure of roofs and floors.		Grade 4: Very heavy damage (heavy structural damage, very heavy non-structural damage) Large cracks in structural elements with compression failure of concrete and fracture of rebars; bond failure of beam reinforced bars; tilting of columns. Collapse of a few columns or of a single upper floor.
	Grade 5: Destruction (very heavy structural damage) Total or near total collapse.		Grade 5: Destruction (very heavy structural damage) Collapse of ground floor or parts (e. g. wings) of buildings.

FIGURE 4.17: Classification of damage to buildings according to EMS-98. The figures are adapted from Grünthal (1998).

LiDAR data is provided by World Bank - ImageCat Inc. (2010b). It has an average density of 3.40 points/m², which is roughly the same as the Mashiki dataset. As we know from Section 4.8, LE-Net is able to operate well when the average density is at least 2 points/m²; therefore, we can expect LE-Net is not influenced by the point density. We obtain the LiDAR data for the study area by extracting it from the original one using the bounding box of the study area (the green rectangle in Figure 4.16). The extracted LiDAR data is shown in Figure 4.18.

Dataset development workflow

In this section, we describe the workflow of developing the Haiti dataset. The objective is to develop a dataset where each point is assigned one of three classes

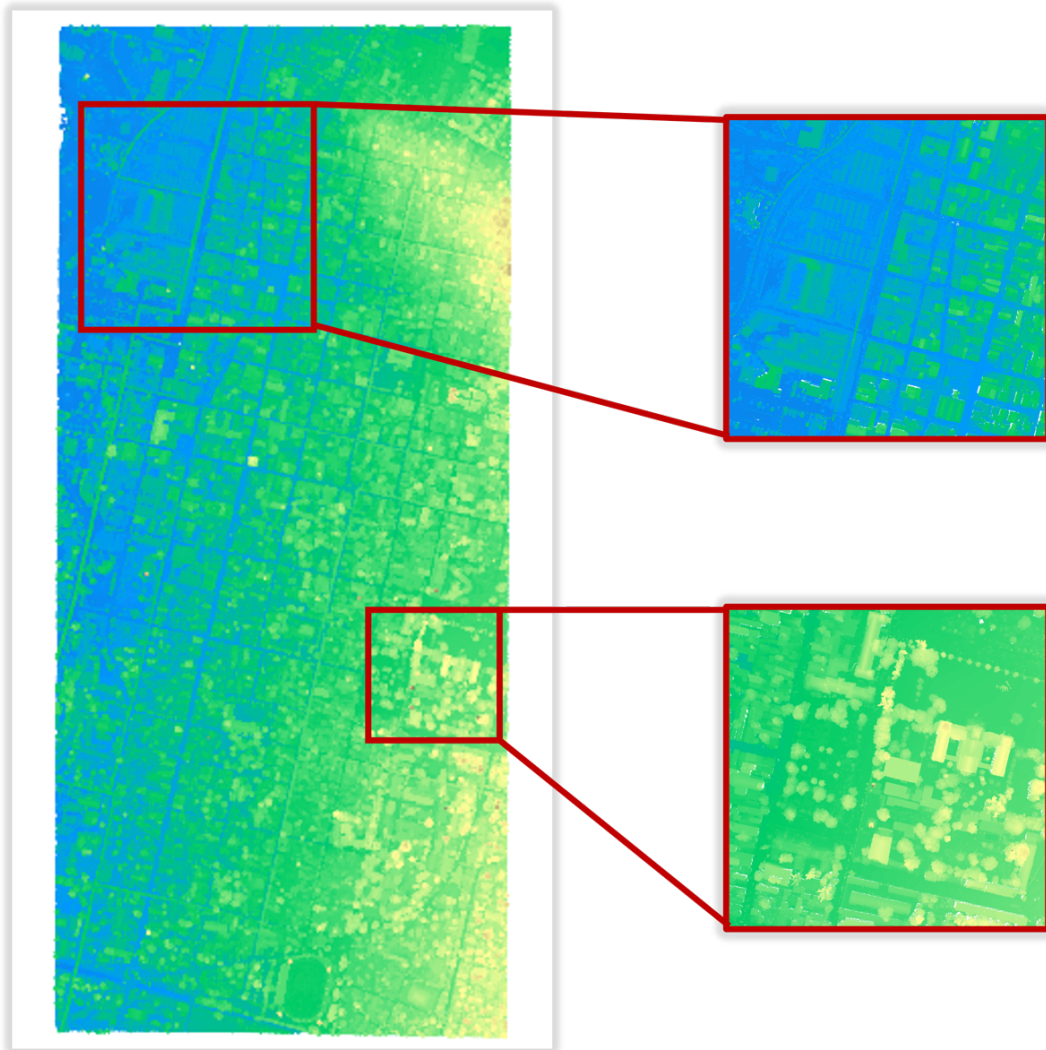


FIGURE 4.18: The LiDAR data used for developing the Haiti dataset. The point clouds are color-coded by elevation values.

including “Collapsed”, “Non-collapsed”, and “Background”. Different from the workflow of the Mashiki dataset, we develop the Haiti dataset in a fully automated manner. Specifically, the entire workflow is split into four successive steps:

- Preclassification
- Footprint overlay and reclassification
- Isolated misclassification refinement
- Building patch extraction

The overview of the procedure is shown in Figure 4.19. In developing the dataset, one big issue is that Grades 1–3 buildings are not included in the visual interpretation result since it only interprets Grades 4–5 buildings. This is problematic because buildings in the study are very densely distributed; thus, a building patch may contain Grade 1–3 buildings, which are not included in the visual interpretation but should be classified as non-collapsed buildings. In order to obtain the labeled cloud efficiently while not ignoring Grades 1–3 buildings, we determine to first preclassify the point clouds into several general classes. These classes include “Ground”, “Non-ground”, “Vegetation”, and “Building”. In particular, the “building” class refers to buildings that are not collapsed. Therefore, the “building” class should include Grades 1–4 buildings (they belong to “Non-collapsed” based on the classification standard used in Chapter 2). In order to improve the quality of preclassification, we compute the height above the ground to augment the preclassification since the height above the ground is known to be useful in airborne point cloud classification (Yousefhusien et al., 2018). In practice, we implement the height above the ground computation and preclassification using LASStools¹. As a result, we obtain the point clouds with the general classification result.

For creating building samples that are suitable for feeding DNNs, we overlay the building footprints from the visual interpretation result onto the preclassified point clouds and overwrite the preclassification result with the damage grade of the building footprints. Specifically, the Grade 4 and 5 buildings are treated as “Non-collapsed” and “Collapsed” classes, respectively, according to Okada and Takai (2000). For reference, we show the damage pattern chart (Okada and Takai, 2000) in Figure 4.20.

¹<https://rapidlasso.com/lastools/>

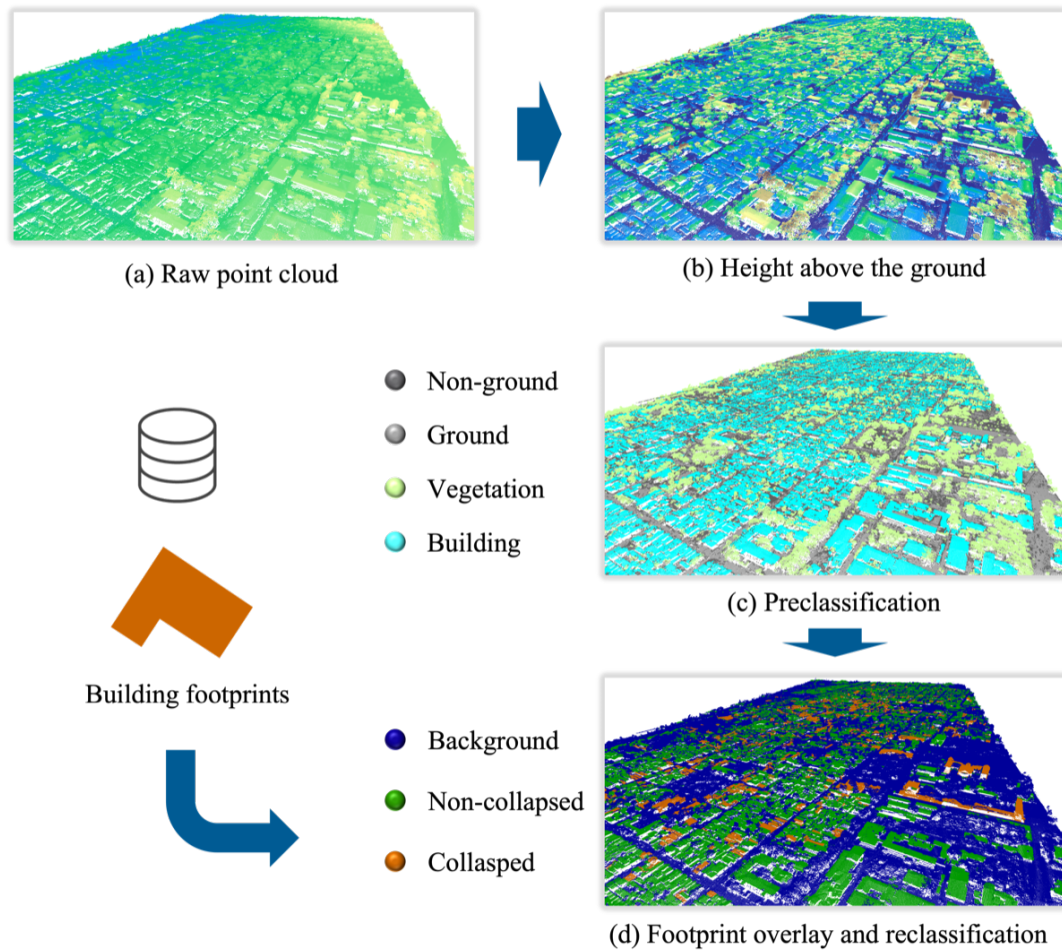


FIGURE 4.19: The overview of the Haiti dataset development workflow. **(a)** the raw point clouds color-coded by elevation values. **(b)** Height above the ground computation. **(c)** Preclassification. Points are classified into four typical classes. **d** Footprint overlay and reclassification. The building footprint is overlayed to the preclassified point clouds followed by the reclassification step. Consequently, each point is classified into one of the three classes (“Collapsed”, “Non-collapsed”, and “Background”).

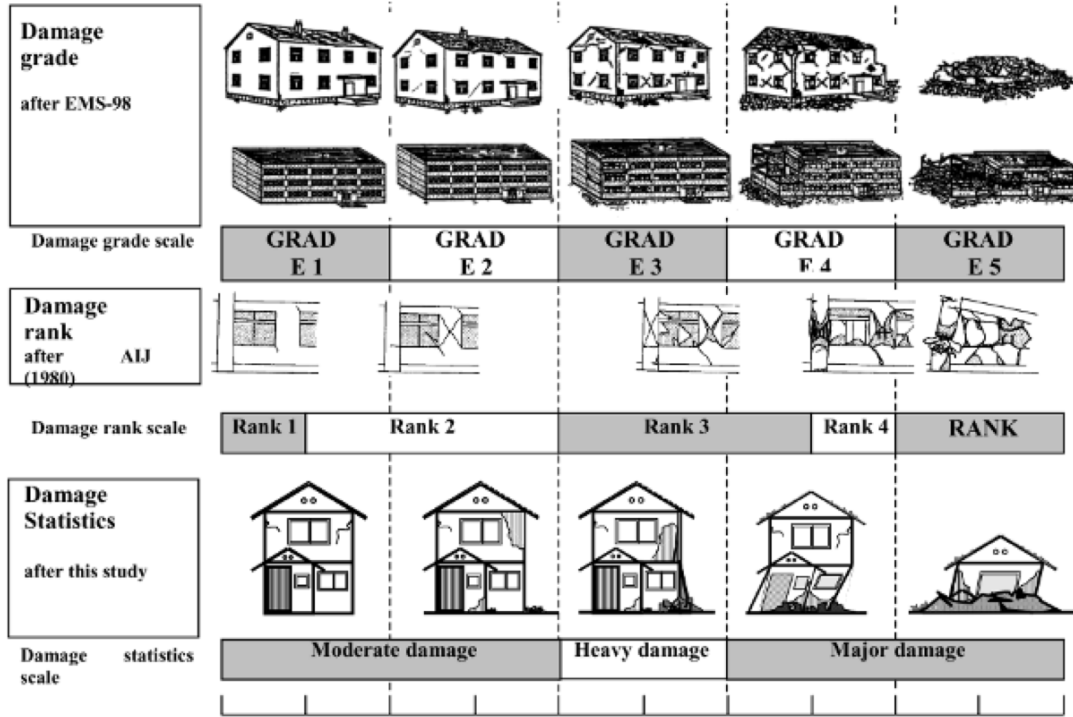


FIGURE 4.20: Comparison among various damage scales (Okada and Takai, 2000)

Then, we perform the reclassification step in which all points belonging to the “Building” class are reclassified as “Non-collapsed” and other points belonging to the “Non-ground”, “Ground”, and “Vegetation” classes are reclassified as “Background”. As a result, we obtain a point cloud dataset where each point belongs to one of the three classes.

The developed dataset is ready to use, however, some isolated misclassifications appear mainly on the edges of buildings due to the coarse annotation procedure. Some typical examples are shown in Figure 4.21. As can be seen, some points are incorrectly annotated due to the inaccurate overlay of the building footprint and point cloud or due to the errors in preclassification. These isolated points may adversely affect the training as well as evaluation; therefore, they need to be removed. To this end, we develop a simple algorithm called the majority filter to refine isolated points from the created ground truth. The algorithm is shown in python-style pseudo code in Algorithm 2. In essence, the majority filter simply replaces the label of the current point with the majority label of its neighbors. For instance, if a point belongs to the “Collapsed” class and its neighbors “Non-collapsed”, the label of the point becomes “Non-collapsed” after applying the majority filter. The effect of the majority filter can be seen in Figure 4.21. As can be seen, some isolated points that appear on the edges of the buildings are reclassified.

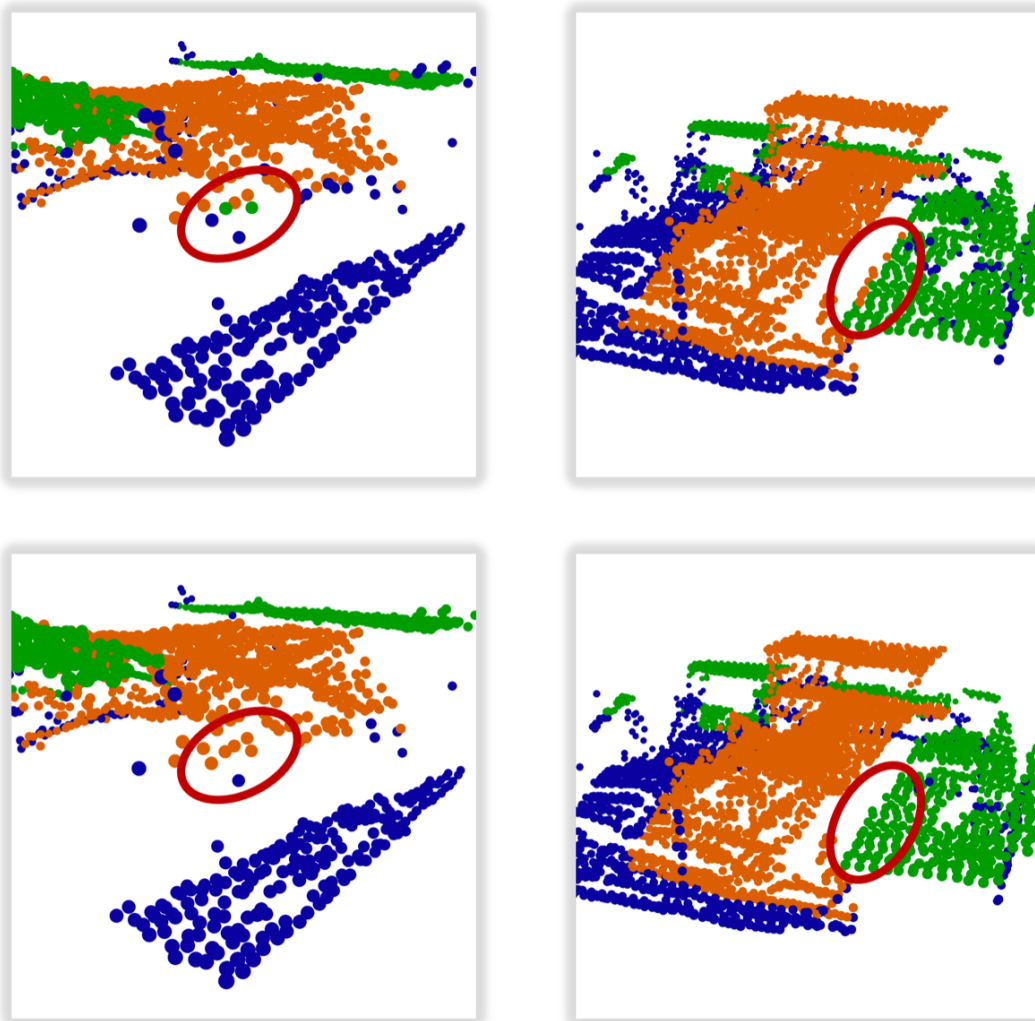


FIGURE 4.21: The effect of the majority filter. After applying the majority filter, the isolated misclassifications are removed after applying the developed majority filter.

Algorithm 2 The majority filter.

```

1: import numpy as np
2: import KDTree
3:  $xyz$  = 3D coordinates of a point cloud
4:  $N$  = the number of points in  $xyz$ 
5:  $tree = KDTree(xyz)$ 
6: Label = the original labels
7: NewLabel = []
8:  $r = 1.5$ 
9: for  $n$  iterations do
10:    $p$  = 3D coordinates of the point  $i$ 
11:    $ind = tree.query\_radius(p, r)$ 
12:   LabelCenter = Label[i]
13:   LabelNeighbors = Label[ind]
14:   u, c = np.unique(LabelNeighbors, return\_counts=True)
15:   SelfIndex = list(u).index(LabelCenter)
16:   if ( $c[SelfIndex] / np.sum(c) < 0.5$ ) then
17:     LabelNeighbors.append(u[np.argmax(c)])
18:   else
19:     LabelNeighbors.append(LabelCenter)
20:   end if
21: end for

```

Statistics and examples

The statistics of the developed dataset are shown in Table 4.6 and 4.7. As we can see, the Haiti dataset contains 2,052 buildings in total where 36% of them are non-collapsed and 64% of them are collapsed. Although the number of buildings in each class looks imbalanced, they have approximately the same number of points according to Table 4.7. Therefore, unlike the Mashiki dataset, the Haiti dataset is somewhat balanced.

Some buildings from the resulting dataset are shown in Figure 4.22. As can be seen, many buildings in the study area are densely distributed. Moreover, the dimension and appearance of buildings vary significantly. For example, the first row of the non-collapsed shows a narrow and long building with a flat roof whereas the bottom row shows a wide building with a gable roof.

TABLE 4.6: The number of buildings in each split in the Haiti dataset.

	Non-collapsed	Collapsed
Train	263	558
Validation	38	167
Test	434	592
Sum	735	1,317

TABLE 4.7: The number of points in each split in the Haiti dataset.

	Background	Non-collapsed	Collapsed
Train	480,384	698,354	655,827
Validation	240,733	211,789	288,074
Test	858,404	945,274	650,679
Sum	1,579,521	1,855,417	1,594,580

The collapse forms of the buildings generally follow the damage patterns described in Schweier and Markus (2006). The first row shows a building that is totally collapsed with many plates remaining on top of the debris. The second row shows a building with a deformed roof. Although the building is still standing, the roof shows a deformation likely due to the structural failure of some stories. The third row demonstrates a typical pattern of RC building collapse where some of the rebars remain vertical whereas other parts are destroyed completely. The collapsed building in the bottom row is one of the most frequent collapse cases observed in the Haiti dataset. In this case, the collapsed buildings are tightly surrounded by non-collapsed buildings. Since they are close to each other, the boundaries between each of them are fuzzy; hence, their separation becomes extremely challenging.

There are substantial differences in terms of building appearances between the Mashiki and Haiti datasets. First, many buildings adopt flat roofs in the Haiti dataset whereas the buildings in Mashiki town adopt gable roofs. This difference is highly influential because the major part of a building captured by the Airborne LiDAR system is its roof. Second, although they share common collapse forms such as “heap of debris” and “pancake collapse”, their collapse forms differ significantly due to the difference in major types of building structures. The main structure type of building in the Haiti dataset is RC and masonry where destroyed buildings are represented as “standing walls without roofs”, “heap of debris with plates”, and so on. Therefore, their collapse forms

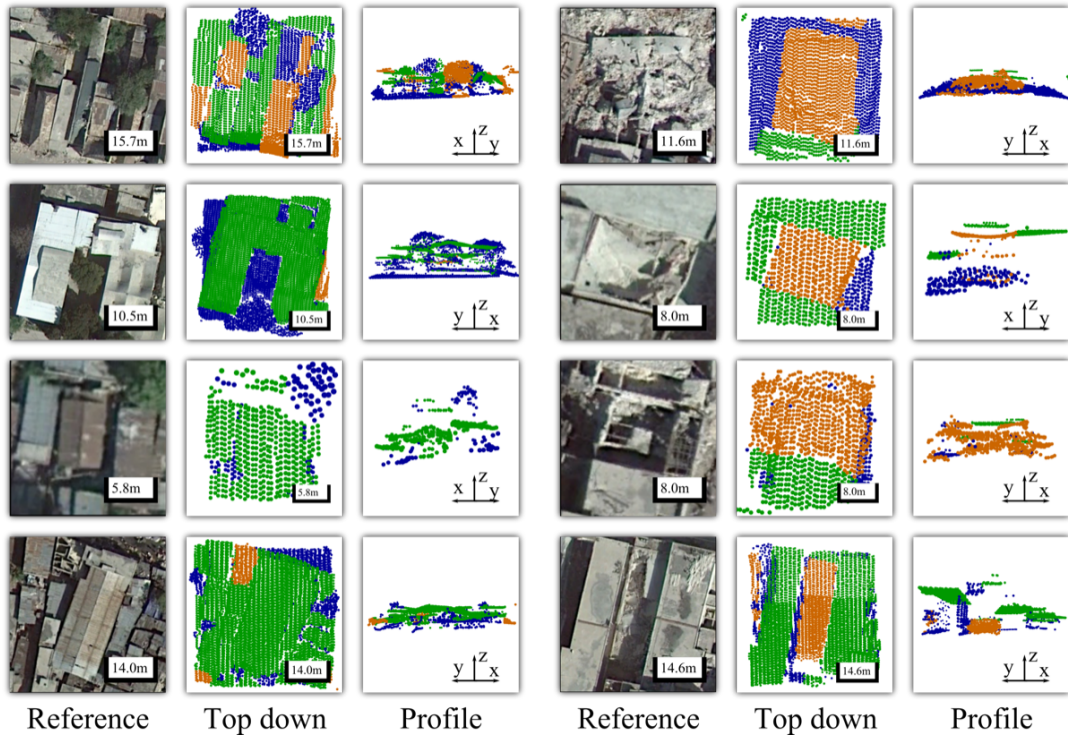


FIGURE 4.22: Visualization of buildings in the Haiti dataset.

Left: non-collapsed buildings. Right: collapsed buildings.

are often represented by the mixture of debris and plates where the roofs are not identifiable. However, the main parts of the roofs of the collapsed buildings in the Mashiki dataset often can be identified even if the entire buildings are collapsed. Lastly, the boundaries of buildings in the Haiti dataset are difficult to be identified because they are extremely close to each other whereas the buildings in the Mashiki dataset can relatively be separated easily.

Differences between the Mashiki and the Haiti dataset

To validate the adaptability and transferability of LE-Net, we need to analyze the difference between the two target regions. Since a LiDAR sensor captures the surface, we limit our discussion to the difference in building appearance. Since LE-Net is designed to separate collapsed buildings from non-collapsed, we analyze the shape of the roof for the non-collapsed building whereas we analyze collapse patterns for comparing collapsed buildings.

Concretely, we randomly take 50 samples from each dataset to visually classified the roof shapes. Although there is no standard for classifying roof shapes, for simplicity, we classified roofs into two classes: roofs consisting of a single plane and roofs consisting of multiple planes. Roofs with a single plane differ

from each other by their sizes and vertical rotations whereas those with multiple planes often present a “umbrella” shape. The illustrations are shown in Figure 4.23 and the classification results are shown in Table 4.8. As can be ob-

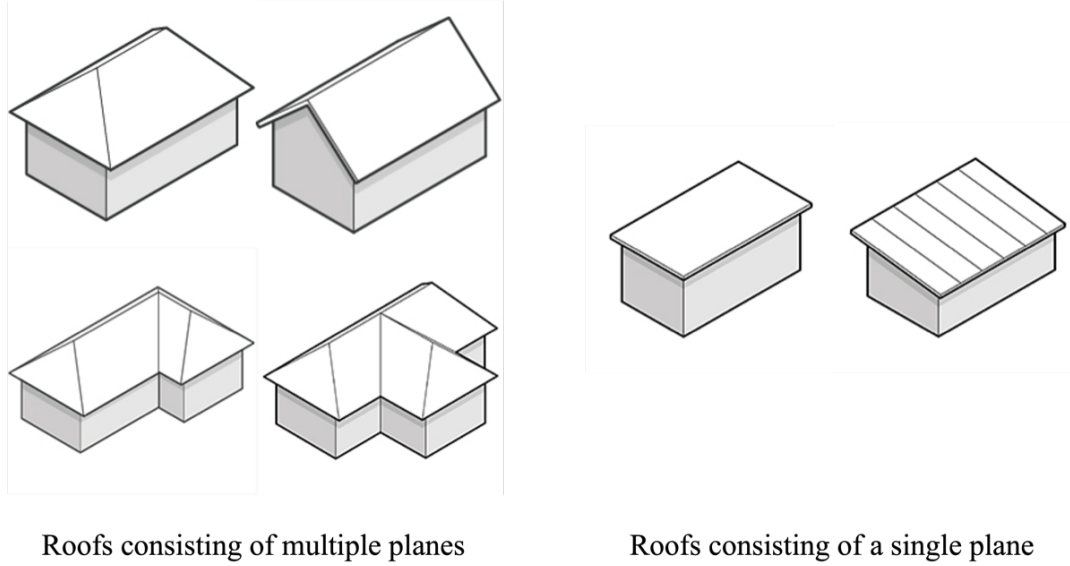


FIGURE 4.23: Roof types classification (JTC ROOFING CONTRACTORS LTD, [n.d.](#)).

Dataset	Roofs with a single plane	Roofs with multiple planes
Mashiki	10%	90%
Haiti	78%	22%

TABLE 4.8: Visual roof types classification result using 50 random samples.

served, the roof type distribution of the Mashiki dataset is significantly different from the Haiti dataset. Therefore, we confirm that the shapes of non-collapsed buildings in the two datasets are very different.

Next, we compare the collapse patterns of buildings in each region. The collapse patterns differ significantly because of the structure type. Thus, we analyze the structure type distributions in the two areas. We obtain statistics from Risai chosa and the Haitian Institute of Statistics and Informatics (more exactly, its English translations from (Eberhard et al., 2010)). Because Mashiki uses the structure type whereas Haiti uses the type of housing as basic units, and because we do not have access to the raw data from Haiti, we present results separately in Table 4.9 and Table 4.10. As can be seen, the major structure type of buildings in Mashiki Town is wooden. In contrast, in the Port-au-Prince region, the main types of housing are one-story house and multistory house or

Type of structure	Wooden	Steel	Concrete and block	unknown
Percentage	82.9%	7.2%	0.7%	9.2%

TABLE 4.9: Structure type distribution of Mashiki Town

Type of housing	Ratio	Roof	Wall	Floor
Kayate (combined roof and wall)	0.3	Thatch or palm leaves or others (52.7)	Earth (74.1)	Beaten earth (93.4)
Taudis or Ajoupas	2.5	Sheet metal (44.7)	Earth (42.6)	Beaten earth (86.7)
One-story house	62.7	Sheet metal (81.9)	Concrete or blocks or stones (75.7)	Concrete (64.0)
Multistory house or apartment	30.3	Concrete (70.9)	Concrete or blocks or stones (96.8)	Concrete (68.6)
Others	4.2	Sheet metal (69.5)	Concrete or blocks or stones (48.2)	Beaten earth (47.6)

TABLE 4.10: Structure type distribution of Port-au-Prince, Haiti. The “Ratio” column shows the ratios of each type of housing in the Port-au-Prince area. The materials for each structure component show the most popular material used for constructing the component for a particular type of housing. The number in the parentheses shows the percentages of such materials among all materials.

apartment which comprises 93% of the type of housing. For these two types of housing, the major material is concrete. Therefore, we confirm that the main structure types of the Mashiki and Haiti datasets are significantly different. Next, we investigate the difference between wooden and concrete buildings in their collapse patterns. The comparison result is shown in Figure 4.24. Qualita-

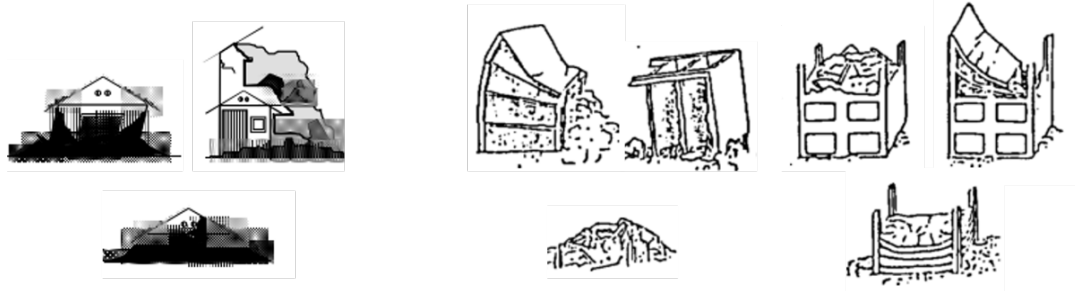


FIGURE 4.24: Comparison of collapse patterns of wooden and concrete buildings (Okada and Takai, 2000). Left: collapse patterns of the wooden building. Right: collapse patterns of concrete buildings.

tively, the typical collapse patterns differ greatly between wooden and concrete buildings. For example, concrete structures may retain vertical pillars vertical even if a building totally collapses, whereas the collapse patterns of the wooden building do not include any such patterns. The only exception is total collapse where buildings totally collapse and become a heap of debris. Therefore, we believe that the difference between the Mashiki and Haiti datasets is significant and thus suitable for validating the adaptability and transferability.

4.3.2 Performance assessment

Evaluation framework and settings

In this section, we perform three-fold evaluations of LE-Net concerning its adaptability and transferability.

First, pretrained LE-Net on the Mashiki dataset is directly applied to the Haiti dataset to quantify the usefulness of knowledge learned from the Mashiki dataset on a substantially different event. By doing this, on the one hand, we can understand how domain differences between two datasets affect the performance in general and can identify what kind of collapse forms can/cannot be detected after training using the Mashiki dataset on the other.

Next, we examine the adaptability of LE-Net by training LE-Net from scratch on the Haiti dataset. Since we have only tested the performance of LE-Net on the Mashiki dataset only, in this part we aim to investigate whether

LE-Net can still perform well on the other dataset with substantially different characteristics.

Lastly, we adapt (retrain) pretrained LE-Net on the Haiti dataset to assess the transferability of LE-Net. Specifically, the LE-Net is first trained on the Mashiki dataset and subsequently trained on the Haiti dataset to assess whether the knowledge obtained from the Mashiki dataset can be transferred well using LE-Net. In addition, since creating the annotation is costly in practice, we also investigate how many training samples are required to achieve certain levels of performance.

As for the experiment settings, we follow the ones used in Chapter 4 except for the downsampling rate when adapting LE-Net to the Haiti dataset. The downsampling rate is set to 4 instead of 2.

Evaluation without adaptation

TABLE 4.11: The result without adapting to the Haiti dataset.
Scores are IoUs (%).

Method	Background	Non-collapsed	Collapsed	Average
LE-Net (Mashiki)	91.1	86.1	50.8	76.0
LE-Net	52.7 (-38.4)	44.8 (-41.3)	22.8 (-28.0)	40.1 (-35.9)
PointNet++	51.6	41.6	21.8	38.3

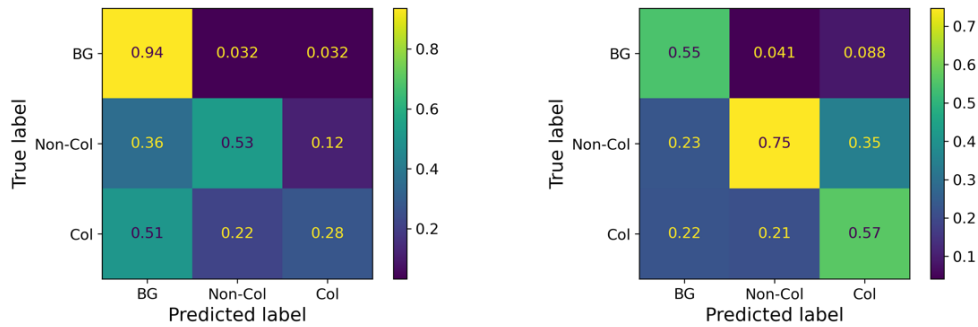


FIGURE 4.25: Confusion matrices of LE-Net without adaptation. Left: the matrix normalized by rows. Right: the matrix normalized by columns.

The result is shown in Table 4.11. In general, the listed methods achieve lower scores compared with the dataset they are trained on. Specifically, LE-Net achieves a 35.9% lower average IoU than the one trained on the Mashiki dataset. Specifically, the “Background” class reduces by 38.2%, the “Non-collapsed” class

reduces by 42.5%, and the “Collapsed” class reduces by 28.0%. Given the substantial reduction of the performance, the impact of domain difference on the performance of the model is very significant. As we can see, the reduction is most severe in the “Non-collapsed” class. We conjecture that this is caused by the huge difference in the appearance of buildings contained in the two datasets because LE-Net learns from the surface geometry. As can be observed from Figure 4.22 and 2.11, they differ significantly in roof shapes, building dimensions, and distributions, which leads to substantially different building shapes and contextual information. Since “Non-collapsed” buildings in the Haiti dataset are also different from both “Non-collapsed” and “Collapsed” in the Mashiki dataset, it is highly likely that they are misclassified as “Background”. This can be verified from the confusion matrix that is shown in Figure 4.25. As can be observed from the left matrix, 36% of the misclassification happens between “Background” and “Non-collapsed”.

Interestingly, the reduction of the “Background” class is also tremendous. In other words, the performance of building localization also degraded significantly. We believe that the reasons are two-fold: 1) the non-collapsed buildings are incorrectly classified as “Background” due to its dissimilarity between the Mashiki buildings and 2) much debris in the Haiti dataset is classified as “Collapsed”, which is excluded and annotated as “Background” in the Mashiki case. As can be seen from the left confusion matrix, the accuracy (i.e., recall) of “Background” is extremely high. In contrast, as shown in the right matrix, the precision of the “Background” is only 55%. The misclassifications between “Non-collapsed” and “Collapsed” are 23% and 51%, respectively, which verifies our observations.

As can be seen in the table, the reduction of “Collapsed” is less severe compared with the reduction of other classes. A possible reason is that the collapse forms in the Haiti dataset are similar to those of the Mashiki dataset to some extent. For instance, the “heap of debris” type of collapse likely appears in both datasets. However, the IoU score of “Collapsed” is significantly lower than those of the other classes. A major reason for this is that much debris is classified as “Collapsed” whereas it is classified as “Background” in the Mashiki dataset. As can be observed from the left matrix, 51% of the collapsed building points are misclassified as “Background”. On the other hand, 22% of “Collapsed” is classified as “Non-collapsed” while 35% of the false alarms are caused by “Non-collapsed”. We conjecture that the fuzziness between Grade 4 and 5 buildings has caused such confusion. According to Okada and Takai (2000), Grade 4 includes partially collapsed buildings, which could be regarded as “Collapsed”.

On the other hand, LE-Net still outperforms PointNet++ in terms of all reported metrics, demonstrating its better generalizability to a different domain.

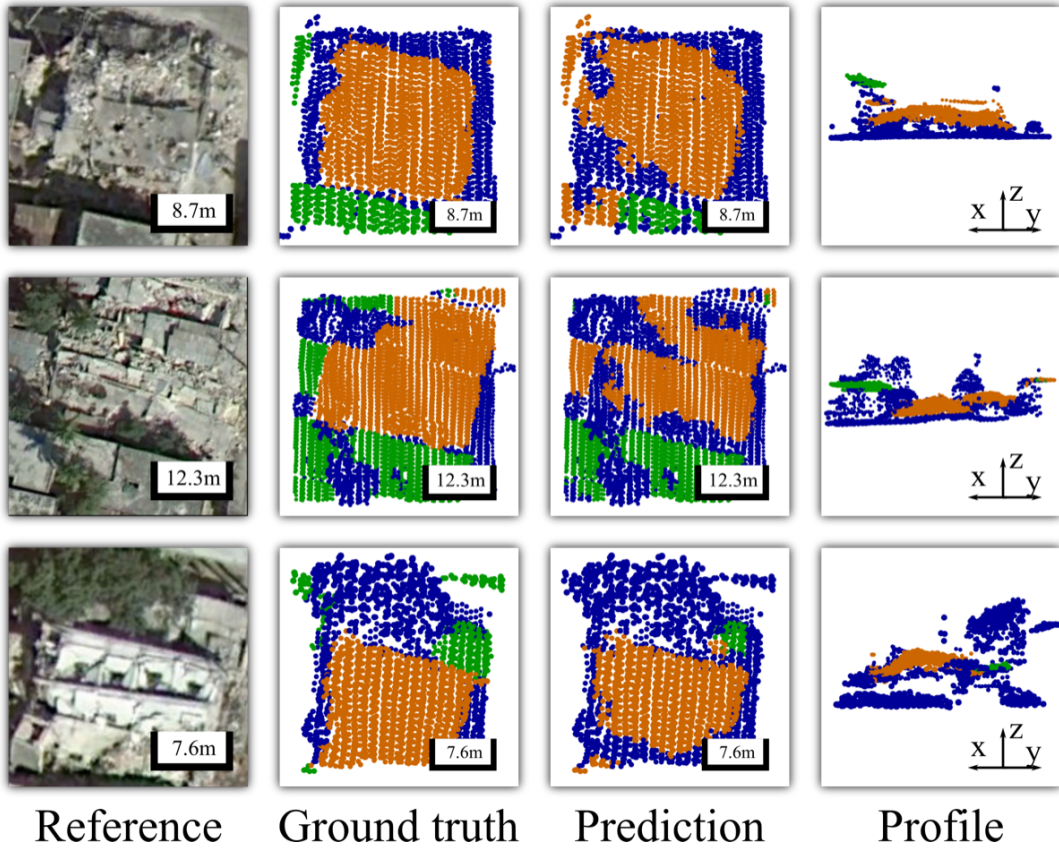


FIGURE 4.26: The qualitative results of LE-Net without adaptation.

We visualize some results to reveal more insights. The results are shown in Figure 4.26. LE-Net pretrained on the Mashiki dataset typically produces acceptable predictions in detecting totally collapsed buildings. For instance, the first row shows a case when a collapsed building is manifested as a heap of debris. We believe that the heap of debris is a common pattern in the Mashiki and Haiti datasets, and thus this type of collapse form can be detected reliably. Moreover, we find that success in detecting this type of collapse can be extended to some specific cases of collapse forms of the Haiti dataset. The examples are shown in the second and third rows. As we can see, the two buildings are both collapsed in a way that the plates withstand in one piece and remain on top of the debris. As shown in Figure 4.27, in fact, this is a common collapse pattern of masonry/RC buildings (Schweier and Markus, 2006) that is not common for wooden-frame structures. We believe that the resulting appearance of this collapse form is similar to that of the heap of debris in general, thus leading to a successful prediction.

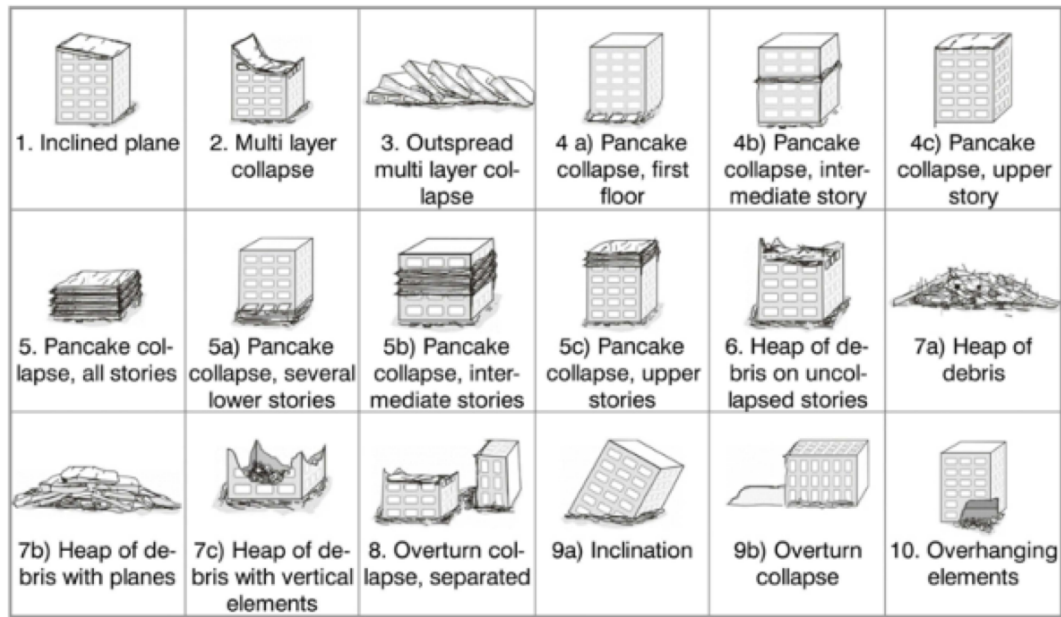


FIGURE 4.27: The compilation of collapse forms. Adapted from He et al. (2016b).

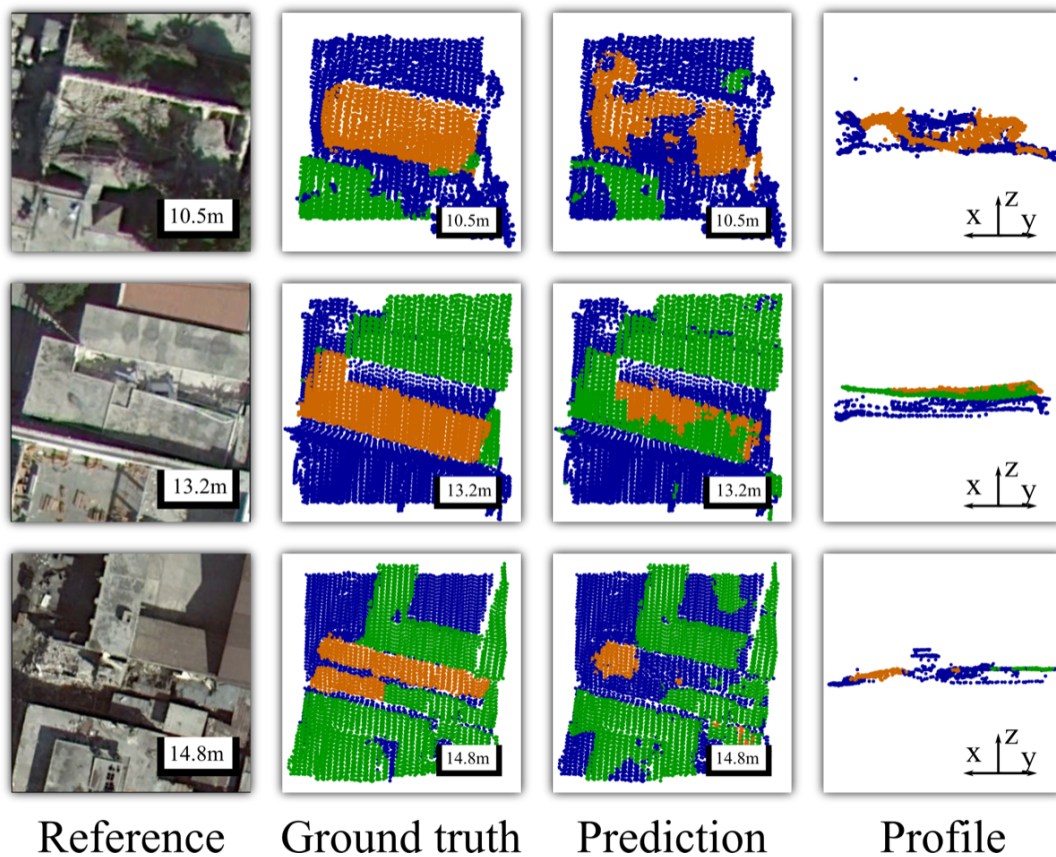


FIGURE 4.28: The failure modes of LE-Net without adaptation.

However, LE-Net fails to detect many types of collapse forms specific to the Haiti dataset. Some typical examples are shown in Figure 4.28. The collapsed building in the first row demonstrates a case where rebars remain above the heap of debris. In terms of appearance, some points are floating above debris, which is a rarely observable pattern for collapsed wooden-frame buildings. As a result, such regions with rebars above are predicted as background whereas the surrounding heap of debris is correctly predicted. Therefore, we presume that LE-Net has difficulty in predicting the specific type of collapse that is significantly different from the Mashiki ones in terms of appearance. However, these types of collapse are often accompanied by debris around, and hence they may be detected in practice by detecting surrounding debris.

The second row also shows a typical collapse pattern in the Haiti dataset. The roof of the target building is significantly deformed probably due to the structural failure of some stories. As we can see from the “Profile” column, the roof smoothly deforms rather than shows non-smooth damage-like patterns. Such a smooth deformation is not present in the Mashiki dataset as it is not a frequent failure pattern of wooden residential buildings. Consequently, we observe significant misclassifications occur on this building where more than 50% of the roof is classified as a non-collapsed building. We conjecture that incorrect predictions are caused by the fact that such smoothly deforming roofs are not frequent in the Mashiki dataset.

In another case, the collapse of narrow buildings is mostly ignored and classified as background (the bottom row). We conjecture that this incorrect prediction is due to the significant difference in contextual information. Specifically, the distribution of buildings in the Mashiki dataset is sparser than in the Haiti dataset. In addition, houses in the Mashiki dataset are often surrounded by background objects like trees. In terms of the building shape, there are few narrow and long buildings in the Mashiki dataset. Therefore, such misclassification is unavoidable due to the above reasons. On the other hand, the left part of the collapse is reliably detected. We believe that this part is not closely surrounded by buildings, and hence the contextual information may be similar to that of the Mashiki dataset. Besides, the heap of debris type of collapse is also present in the Mashiki dataset. Therefore, LE-Net manages to produce correct predictions on this part.

Evaluation of the adaptability

The quantitative results are listed in Table 4.12. In general, LE-Net successfully outperforms PointNet++ in terms of average performance in each class, which

TABLE 4.12: The result without pretraining. Scores are IoUs (%).

Method	Background	Non-collapsed	Collapsed	Average
PointNet++	65.9	62.6	38.8	55.8
LE-Net	68.4	64.4	40.8	57.9

is consistent with the case of the Mashiki dataset. Therefore, LE-Net has better adaptability than PointNet++. Interestingly, the improvement is not limited to the “Collapsed” class and improves by about similar numbers. Therefore, we believe that LU can work well in detecting many other classes that benefit from the enhancement/suppression of discontinuities.

In particular, an approximately 2% improvement over PointNet++ is observed in detecting the “Collapsed” class. Therefore, LE-Net is better at adapting to the dataset that has significant collapse forms. Although LE-Net’s design is motivated by the prior knowledge observed from the Mashiki dataset, we believe that the prior knowledge is sufficiently general and can be generalized to different collapse forms such as the ones that appear in the Haiti dataset.

Further, the improvement is most conspicuous in detecting background classes (+2.4%); therefore, LE-Net is especially good at separating background and buildings. In other words, it performs better than PointNet++ in building localization. This is an important characteristic especially when the building footprints of the affected regions are not available. Furthermore, a 1.8% improvement is observed in detecting the “Non-collapsed” class. We believe that LE-Net’s ability to explicitly manipulate smoothness information represented by the discrete Laplacian is also useful in detecting non-collapsed buildings since most buildings in the study area adopt flat roofs.

To reveal more insights about how LE-Net improves PointNet++, we visualize some results in Figure 4.29. For instance, the first row shows a common case in the Haiti dataset where a relatively small collapsed building is located in between other buildings. Although PointNet++ almost fails to detect it, LE-Net locates the building accurately. We think that the ability to enhance/suppress the discontinuities increase the sensitivity of LE-Net to the collapsed buildings, leading to better detection.

LE-Net also adapts well to a situation specific to the Haiti dataset. In the second row, the roof of the collapsed building might fall off or be destroyed while its walls are still standing, exposing the structure of the building. PointNet++ completely fails to detect this collapsed building by predicting them as “Background”. Therefore, it is not trivial to detect this type of collapse. In contrast,

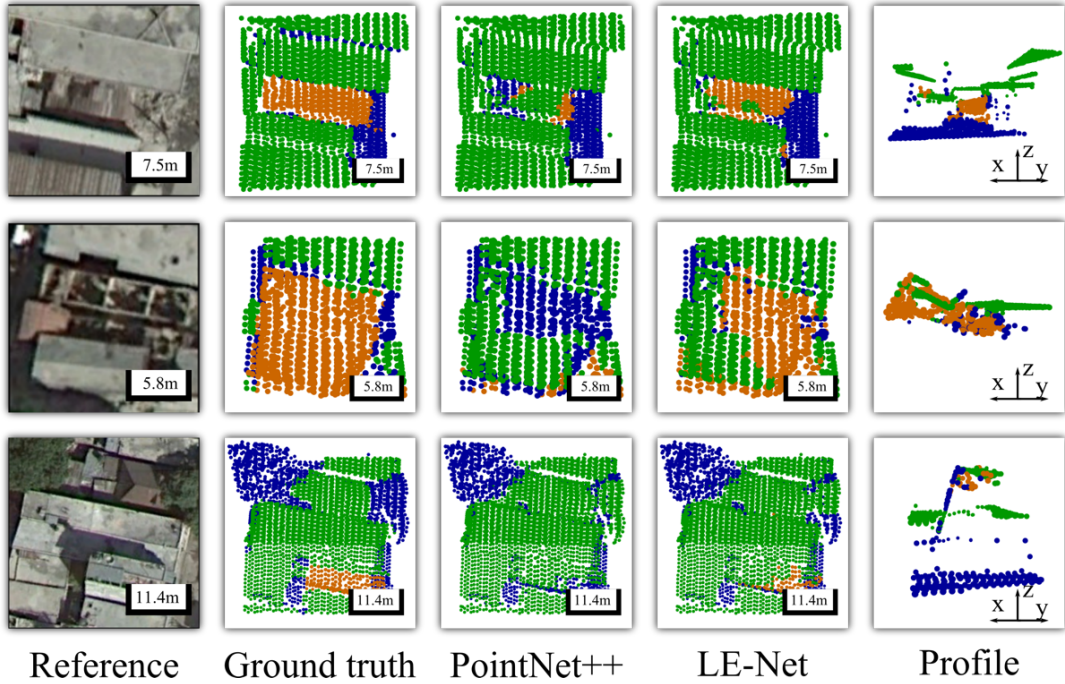


FIGURE 4.29: The qualitative results of the adaptability experiments.

the enhanced capability of LE-Net enables LE-Net to detect it successfully. Since the major difference between LE-Net and PointNet++ is LU, we believe that LU’s ability to enhance such discontinuities leads to the enhancement of the feature representation of this type of collapse pattern.

Furthermore, the bottom row shows a challenging case where an inclined and tall building is among the lower buildings, densely distributed. This pattern is rare even in the Haiti dataset, and thus detecting such a kind of collapse may be difficult. As expected, PointNet++ ignores this collapsed building completely. On the contrary, LE-Net manages to produce partially correct predictions. This is significant in practical situations because it avoids a possibly fatal omission error. As can be seen, the correctly predicted part is a heap of debris appearing on the building. We believe that it is successfully detected because of the selective enhancement brought by LUs.

Evaluation of the transferability

The result is shown in Figure 4.30 and Table 4.13. As can be seen, the effect of pretraining, which is the effect of knowledge obtained from the Mashiki dataset, is most obvious when available training samples are few. Specifically, the average IoU (mIoU) of LE-Net is substantially higher (+12.6%) than the one without pretraining when only 2.5% (21 building samples) of the training

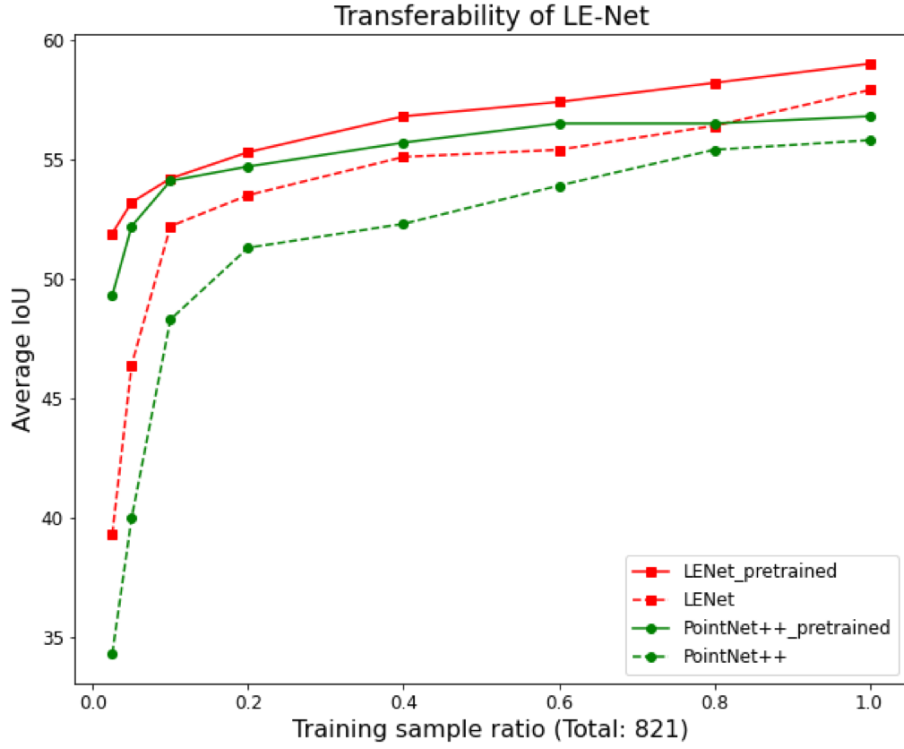


FIGURE 4.30: The transferability of LE-Net with varying numbers of training samples.

TABLE 4.13: The effect of pretraining with increasing numbers of training samples (%). The percentages in the parentheses indicate the amount of training data used for adaptation.

Method	21 (2.5%)	41 (5%)	82 (10%)	164 (20%)	328 (40%)	493 (60%)	657 (80%)	821 (100%)
LE-Net	39.3	46.4	52.2	53.5	55.1	55.4	56.4	57.9
LE-Net (pretrained)	51.9	53.2	54.2	55.3	56.8	57.4	58.2	59.0
Δ	12.6	6.8	2.0	1.8	1.7	2.0	1.8	1.1

samples are available. In practice, making annotations is always an expensive procedure that requires a network of experts to interpret and handle data. Therefore, obtaining substantially better performance with only a small fraction of data is an attractive trait.

Moreover, it is observed that the gap in the scores between pretrained LE-Net and LE-Net without pretraining gets smaller with increasing training samples. Therefore, the effect of pretraining is especially useful when training samples are few and its effect gradually diminishes when the number of training samples increases. We believe that with the increasing number of training samples of the application domain, the most suitable feature representation can be learned quickly and replace those of the sub-optimal ones that come from the dataset that performed pertaining. Interestingly, despite the gap between pretrained and non-pretrained models becoming smaller with increasing training samples, we observe that this gap is initially made small quickly and stabilized from about 20% training sample used. Therefore, we conjecture that although much knowledge is replaceable, some knowledge useful knowledge from the Mashiki dataset is not replaceable and keeps being effective even when the training samples keep increasing rapidly.

On the other hand, we observe that LE-Net, regardless of their use of the Mashiki dataset, outperforms PointNet++ consistently. In fact, LE-Net outperforms the pretrained PointNet++ when 100% of the training data is used. Therefore, we believe that LE-Net has better adaptability and transferability compared with PointNet++.

4.3.3 Analysis and discussion on the performance gap between the Mashiki and Haiti results

Comparing the result of CBD in the Mashiki dataset (Section 4.2.1) and the result of the adaptability, we find that there is an approximately 18% mIoU gap between the result of the Mashiki and Haiti cases. Therefore, here we provide an analysis to identify the possible reasons for such a gap.

First, we reduce the training data in the Mashiki to match that of the Haiti case while leaving test data unchanged. Then we retrain LE-Net using similar numbers of training data and perform predictions. The results are reported in Figure 4.31. As can be seen, we observe systematic differences between the performance of LE-Net trained on the Mashiki and Haiti datasets. Therefore, the number of training data may not be the direct cause of the performance gap.

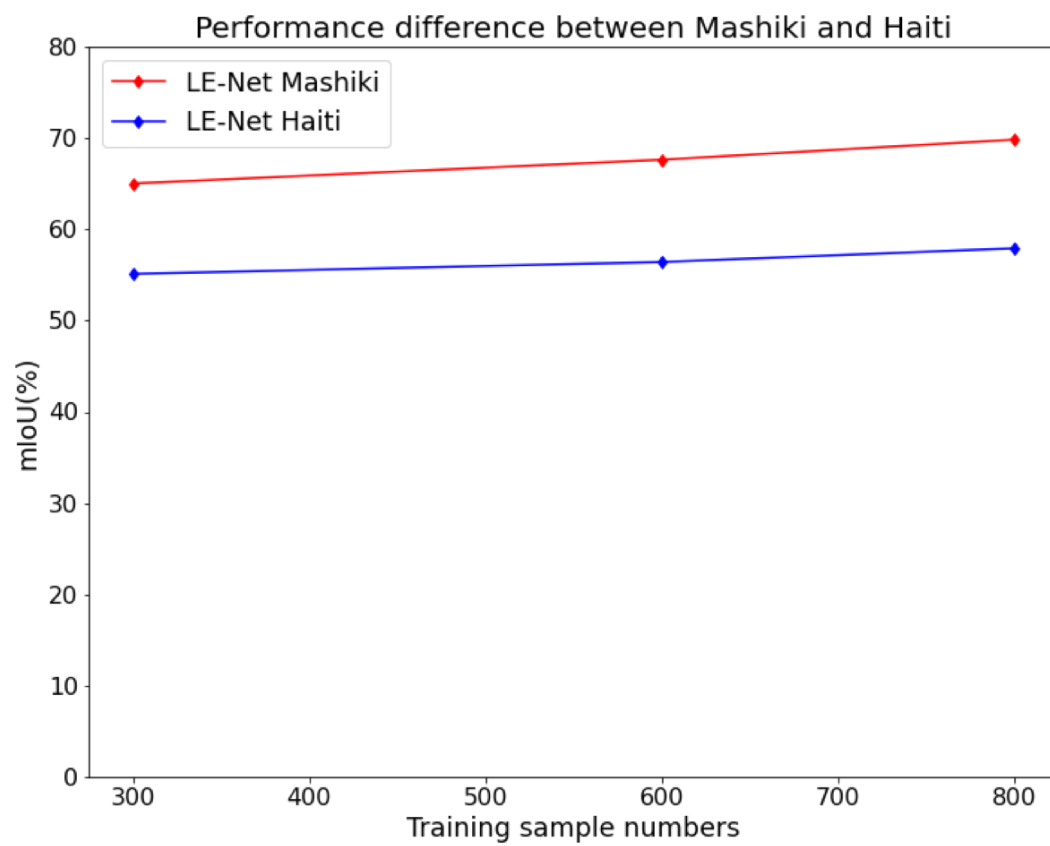


FIGURE 4.31: Performance difference between the Mashiki and Haiti results with varying numbers of training samples.

On the other hand, we find that there are ambiguities in the definition of non-collapsed and collapsed buildings in the Haiti dataset. The ambiguities are illustrated in Figure 4.32. As can be seen from the figure, there are

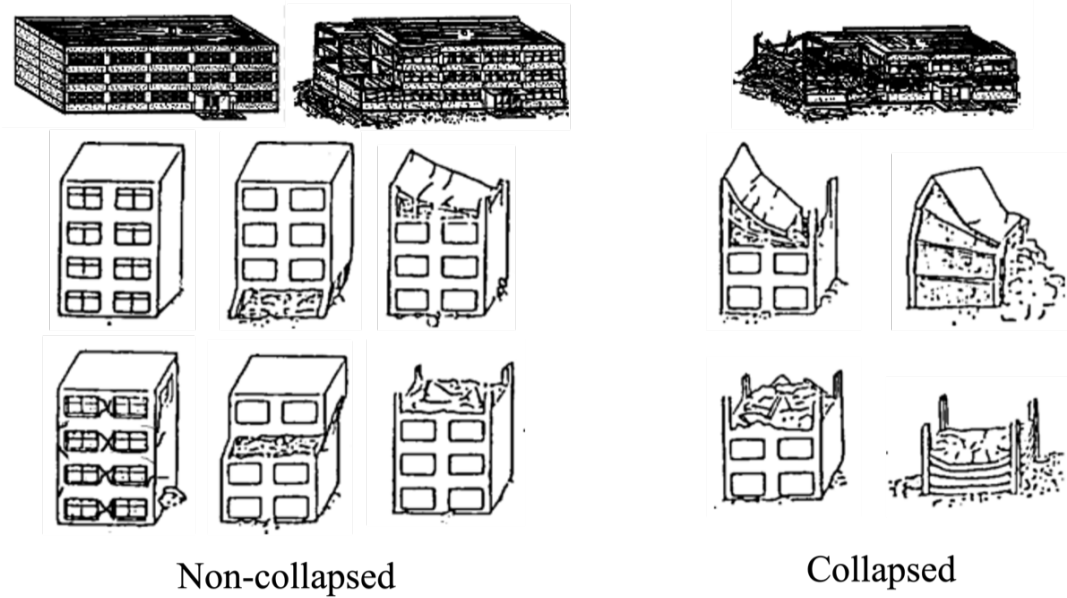


FIGURE 4.32: Ambiguities in the definition of non-collapsed and collapsed buildings in the Haiti dataset. The figure is adapted from Okada and Takai, 2000.

ambiguities in both definitions of non-collapsed and collapsed buildings. Concerning non-collapsed buildings, the most severely damaged buildings would appear significantly different from others in the LiDAR sensor. On the other hand, they would appear very similar to the collapsed buildings instead. Such an ambiguity undoubtedly brings classification difficulty to human operators who often visually interpret the building damage from aerial images or LiDAR data. Moreover, it is very challenging for a model to learn from such data and perform well. Therefore, we believe this ambiguity may be the major cause of the decrease in performance compared with the Mashiki case. To verify this point, we show some examples reflecting such ambiguities in Figure 4.33. As we can see, the top row shows a case where a building with a significant inclination is classified as “Non-collapsed”. On the contrary, the bottom row shows a case where a building is classified as “Collapsed” even though most part of its roof is nearly undamaged. Such inconsistency in the ground truth definition surely causes confusion and leads to degraded performance. To remedy this, we conjecture that a LiDAR-specific damage chart needs to be introduced so that buildings can be more easily separated. In addition, we also find some other problems related to ground truth: 1) roofs are inclined/deformed but not

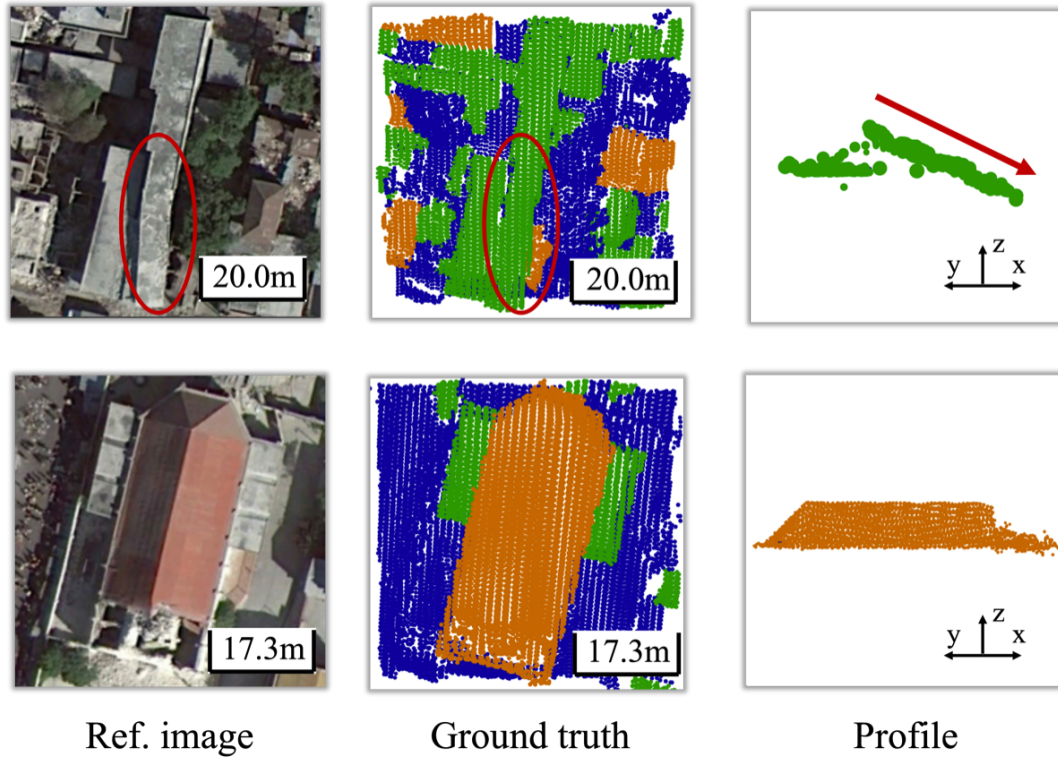


FIGURE 4.33: Examples of ambiguities in the Haiti dataset.

observable from aerial images; 2) small and narrow buildings that cannot be recognized from aerial images; 3) building shadows that hide target buildings.

However, we note that LE-Net manages to learn useful patterns from such a dataset which is validated by the adaptability and transferability experiments; hence, we believe that sufficient performance is achieved and the validation of the adaptability and transferability are achieved.

4.4 Summary

In this section, the effectiveness, design choices, decision-making, limitations, efficiency, and robustness of LE-Net are validated using the Mashiki dataset. Moreover, the adaptability and transferability of LE-Net are validated.

- **Effectiveness:** The quantitative evaluation reveals that LE-Net outperforms those methods significantly in terms of overall performance. Then, the detailed comparison of LE-Net and PointNet++ demonstrates that LE-Net consistently outperforms PointNet++ in terms of precision, recall, and IoU. In particular, LE-Net performs exceedingly well on the “Collapse” class; therefore, the effectiveness of the LE-Net in detecting collapsed buildings is successfully validated. In addition, the quantitative

analysis verifies that LE-Net especially not only performs well in detecting buildings with severe visible damage but also can detect the buildings that do not present significant clues of damage (e.g., a slight inclination), which tend to be ignored by PointNet++.

- **Design choices:** To understand the role of LU better, we also perform various analyses of LU. First, we conduct the component analysis and verified that each component of LU is vital for achieving the best performance. Next, we visualize the most influential part, the function $\mathcal{F}$, to investigate its behavior. As a consequence, we understand that $\mathcal{F}$ can enhance feature representations of relevant discontinuities (e.g., roof deformations) and suppress the irrelevant ones simultaneously.
- **Decision-making and limitations:** We also investigate the decision-making of LE-Net through visual explanation, which is important to applications like CBD. The result shows that LE-Net bases its decisions on debris, roof deformations, and regions where roofs and ground touch. Furthermore, we identify some typical failure modes of LE-Net. We identify that LE-Net has difficulty in inferring the structural consistency of the building when it is broken apart. Moreover, LE-Net also tends to predict the near-complete part of the building as “Non-collapsed” and the collapsed part “Collapsed”. We also confirm that it is challenging to recognize a pancake collapse well due to the lack of roof deformation or inclinations.
- **Efficiency:** We analyze the computational efficiency of LE-Net. It reveals that although LE-Net requires slightly more parameters than other networks, it does not overfit. More importantly, LE-Net requires the fewest FLOPs compared to others, which means it is more efficient in the use of computation resources. Therefore, LE-Net is suitable for real applications where computation resources are limited (e.g., emergency damage mapping).
- **Robustness to the density variation:** we investigate the robustness of LE-Net w.r.t the point density. It has been shown that LE-Net performs well with about 60% of the original average density.
- **Adaptability:** The adaptability of the LE-Net is validated using the Haiti dataset. The comparison of LE-Net of PointNet++ reveals that LE-Net consistently outperforms PointNet++ in all evaluation metrics.

- **Transferability:** The transferability of the LE-Net is validated by adapting pretrained LE-Net and PointNet++ to the Haiti dataset with varying numbers of training samples. As a result, we find that pretrained LE-Net always outperforms PointNet++; hence, the better transferability of LE-Net is verified. Further, we observe that the knowledge from the Mashiki dataset is most useful when available training samples are few, which is often the case in emergency damage mapping. In addition, although the usefulness of the knowledge from the Mashiki dataset diminishes with increasing numbers of training samples from the Haiti dataset, pretrained LE-Net always outperforms the non-pretrained counterpart. The same pattern is also observed in the PointNet++ case. Therefore, some of the knowledge learned from the Mashiki dataset is general and have a long-term positive effect even with the training samples keep increasing.

Chapter 5

Conclusions

5.1 Summary

CBD is important for humanitarian rescue and obtained information is useful for post-disaster recovery. Motivated by the fact that 3D point clouds are suitable for CBD, and motivated by the remarkable performance of DNNs in point cloud analysis, this research has explored the possibility of deep learning-based CBD using 3D point clouds. Based on the extensive literature review, we identify two issues that prevent this research from going further: 1) the lack of a suitable dataset on which DNNs can be trained and evaluated and 2) the lack of a method dedicated to CBD. The first issue is a fundamental one because a well-prepared dataset has a tremendous impact on the performance of DNNs. The second issue reveals that the domain-specific knowledge of CBD is yet to be integrated into the model design of DNN. Therefore, we set two goals for this research, which are:

- Objective 1: Develop a suitable dataset for point cloud-based CBD and analyze the performance of existing DNNs,
- Objective 2: Develop a dedicated method for CBD by integrating domain-specific prior knowledge into the model design.

We choose the 2016 Kumamoto earthquake as our target event. The field survey results and post-event LiDAR data of the Mashiki town, the most severely affected region, are used for developing the Mashiki dataset. A seamless development workflow is designed for this purpose. The benchmarking of several mainstream DNNs is performed to investigate what can and cannot be done by DNNs with a special focus on the analysis of PointNet++, the standard network for 3D point cloud processing. The result reveals that buildings that show severe signs of damage can be detected reliably but those with insignificant clues cannot be detected or even missed. We conclude that the most practical way to detect such collapsed buildings is to use prior knowledge of damage.

In order to develop a method dedicated to CBD by integrating domain-specific knowledge into the model design, we first perform an in-depth analysis regarding the nature of the damage and identify the usefulness of smoothness information. To encode smoothness while enabling a natural integration to DNNs, we choose the discrete Laplacian as our fundamental tool, and its usefulness to represent damage numerically is subsequently verified. We also identify a problem of the discrete Laplacian, that is, it lacks selectivity in differentiating the useful discontinuities from the less useful ones. To this end, we propose LU, a computational module that can encode smoothness information selectively by enhancing the useful one and suppressing the less useful one. Using LU as a core, we construct LE-Net, a network dedicated to CBD.

The effectiveness of LE-Net is validated using the Mashiki dataset. Specifically, LE-Net outperforms all other listed baselines in the CBD task. It can detect buildings with better accuracy with fewer false alarms. Specifically, LE-Net can detect buildings with insignificant damage clues, that is, buildings without significant roof deformation and with slight inclinations, which are the buildings that are completely missed by PointNet++. In addition, various analysis of LU is performed to understand its effect quantitatively and qualitatively. The analyses verify that each component of LU is vital with the selectivity function being the most influential factor. Therefore, its roles are also investigated in detail and verified its selectivity qualitatively. Given the importance of interpretability in CBD, we perform visual explanation analysis to investigate the decision-making of LE-Net. As a result, we find that LE-Net makes classification decisions by accurately locating damage-related regions. The efficiency and robustness of LE-Net are also investigated. Further, the applicability of LE-Net under various scenarios is also validated.

We also examine the adaptability of LE-Net to a substantially different dataset and its transferability which is its ability to reuse the knowledge learned from the Mashiki dataset. For this purpose, we develop another dataset, termed the Haiti dataset, which is developed using data acquired after the 2010 Haiti earthquake. By training the models on the Haiti dataset without pretraining on the Mashiki one, we verify that LE-Net has better adaptability compared to PointNet++. Also, the transferability of LE-Net is also verified. Pretrained LE-Net consistently outperforms the PointNet++ when increasing the number of training samples gradually. The knowledge from the Mashiki dataset is most useful when training samples of the Haiti dataset are few and it diminishes with increasing training samples; however, we find that the effect of pretraining, though it diminishes, seems to have a long-term positive effect.

We provide the answers for the above objectives in the following.

Answer to Objective 1: We have achieved the objective. We have created the Mashiki dataset and benchmarked several mainstream DNNs. The Mashiki dataset is developed in a rigorous way and enables a comprehensive evaluation of DNNs. The benchmarking result indicates that the mainstream DNNs cannot achieve satisfactory performance. Therefore, further effort needs to be spent on developing a dedicated method for CBD.

Answer to Objective 2: We have achieved the objective. Based on the analysis of damage characteristics, we develop LU that selectively enhances/suppresses discontinuities. The LE-Net is constructed using LU as a core operator. Extensive experiments have verified the effectiveness of LE-Net and confirmed the usefulness of LU.

We believe that the contributions made by this research have a significant impact on advancing the DNN-based CBD using 3D point clouds research. Specifically, the developed workflow can be a reliable standard when developing another dataset. How we develop LE-Net can be a nice conceptual framework for integrating prior knowledge into the model design for CBD. Moreover, the results of this research can be used directly in real-world applications. Given the Mashiki dataset, we can expect LE-Net to work well if the characteristics of the target area are similar to the Mashiki town and the event is an earthquake. The developed dataset is also helpful as a pretraining dataset so that it can effectively improve the application of a model to a different region, especially when training samples are few. Given the effectiveness, efficiency adaptability, and transferability of LE-Net, it can be the default choice that a user can adopt.

5.2 Limitations

We describe the most significant limitations found in this research below.

- Our specific choices of datasets (Mashiki and Haiti) inevitably introduce some limitations to this study. Specifically, first of all, we cannot validate comprehensively the effectiveness of LE-Net in detecting collapsed buildings other than wooden and concrete buildings. For instance, we cannot comprehensively verify LE-Net’s effectiveness in detecting steel-framed buildings. Secondly, LE-Net’s performance may also be greatly influenced by roof shapes that are not present in the Mashiki or Haiti since it learns from geometry.

- Although much automation is realized during the development of a point dataset for deep learning-based CBD (e.g., Section 2.3.1, 2.3.3), we generally find that the development workflow is still time-consuming. Most of the inefficiency comes from the annotation process of point clouds. The annotation needs to be done manually and the manual selection of building point clouds is not trivial and requires domain expertise. Besides, manual selection itself takes much time; hence, manual selection needs to be further assisted and automated.
- LE-Net cannot infer the structural consistency of the collapsed building. For example, it is shown that LE-Net has difficulty in recognizing partially collapsed buildings (Section 4.2.3). We observe a typical failure pattern where LE-Net predicts the collapsed part of the building correctly but fails to identify the standing part. A possible reason is that the shape of the standing part is relatively complete like a non-collapsed building. Also, this point is also validated by Section 4.2.3 where LE-Net cannot identify the pancake-collapsed building due to the completeness of the roof shape even though it correctly identifies the closely located deformed roof.
- LE-Net’s excessive dependence on the local geometry. As can be seen from Section 4.2.5, LE-Net cannot outperform PointNet++ when the average point density is extremely low. We think that the loss of fine geometry influences the quality of the discrete Laplacian greatly, leading to incorrect geometric representation, which in turn adversely affects the performance.

5.3 Future Work

Some probably fruitful research directions that extend this research are identified as follows:

- More efficient point cloud annotation. As shown in Section 4.3.1, suitable automation in preclassifying the point cloud greatly reduces the annotation cost. We think that a feasible way to speed up the annotation is to create a better preclassification classifier with a small number of training samples and a large number of hand-crafted features. The key to this is that some machine learning classifiers are more data efficient than DNNs. They can be trained well with fewer training samples as long as the hand-crafted features are well-designed. However, the suitability of hand-crafted features needs to be identified and machine learning classifiers need to be compared. In addition, the performance of the classifiers

w.r.t the number of training samples needs to be analyzed in order to quantify annotation-automation trade-offs.

- More label noise tolerable training/methods. Although creating a dataset with very accurate labels is difficult, developing a dataset with coarse annotations is not time-consuming as long as the locations of buildings are known, as is demonstrated in Section 4.3.1. The noisy labels are relatively easy to produce by first locating buildings and then annotating the buildings simply (e.g., annotating buildings with simple bounding boxes). One way to improve the performance without an accurate dataset is to train the models with label noise robustness (Song et al., 2022). Since the primary goal of CBD is to detect not to detect perfectly, the noisy labels can still be useful as long as the model is informed about how noisy the training samples are.
- Make LE-Net more structure-consistency aware. As described in Section 5.2, LE-Net is not good at recognizing partially collapsed buildings due to the completeness of the non-collapsed part of the buildings. One way to resolve this is to change the task formulation so that a building can only have one label. For example, we formulate the task as instance segmentation so that a building can only have one label. In addition, we can also formulate the task as joint classification and segmentation. Similar to the above solution, in this way we can force the model to predict a category label for a building. On the other hand, from the methodological point of view, we can extend LE-Net in such a way that it can exploit the relationship between identified buildings and their surroundings better. For example, we can deliberately make the predictions more consistent by enforcing prediction smoothness based on some proximity measure. Furthermore, the difference in predictions is caused by the difference in feature representations; hence, feature representations within partially collapsed buildings can be made close by methods like contrastive learning.
- Make LE-Net more robust to the density variation. LE-Net inevitably is vulnerable to the corruption of local geometry because of the use of Laplacian. Therefore, some measures can be taken to tackle this vulnerability. The most direct way is to use data augmentation techniques so that the model can learn to be robust. A typical data augmentation technique is random point dropping (Qi et al., 2017b) which drops points with some probability before feeding point clouds to the model. For another example, we can implement training with dynamically changing scales (Xiu,

Shinohara, and Matsuoka, 2019). In this method, the input is dynamically extracted with varying radii from the entire point cloud on-the-fly. When extracting, the points are randomly sampled; therefore, the same building may produce a point cloud with a different average density and a different amount of contextual information. From a methodological point of view, the neighborhood selection process can be improved such that LU only finds reliable neighbors. The currently used method for neighbor point selection is kNN which is a deterministic method. Therefore, it is directly affected by the corruption of local geometry. To combat this, we can design a learnable neighbor point selection so that LU has enhanced flexibility in choosing reliable and effective neighbors.

Appendix A

Additional Analyses

The content of this chapter is mainly based on Xiu et al. (2023).

A.1 Extended Comparison and LE-Net Large

TABLE A.1: Extended comparison of overall performance. OA (%) and mIoU (%) of each method are reported. The bold texts indicate the best performance. Param. (M) and FLOPs (G) denote the number of parameters and floating point operations per second, with lower scores indicating higher efficiency.

Method	Param.	FLOPs	OA	mIoU
PointNet (Qi et al., 2017a)	3.5	7.2	90.6	71.5
PointNet++ (Qi et al., 2017b)	3.5	8.6	92.3	72.8
DGCNN (Wang et al., 2019b)	1.6	5.4	90.3	72.0
UPA (Xiu et al., 2021a)	6.6	11.9	93.1	74.8
PAConv (Xu et al., 2021)	11.0	61.7	93.8	74.9
PointMLP (Ma et al., 2022)	6.4	14.1	93.5	75.1
LE-Net (Ours)	4.3	4.1	93.4	76.0
LE-Net Large (Ours)	6.3	9.7	94.5	80.1

Resulting LE-Net is an efficient network that only requires a small number of computational resources. This trait makes LE-Net especially fitting to emergency disaster response where collapsed buildings need to be detected with limited computation resources. On the other hand, it is also important to explore the potential of LE-Net given sufficient computation resources by which the detection accuracy can be effectively improved, thereby offering users an opportunity to freely choose whether to prioritize efficiency or target accuracy based on practical requirements and available resources. To this end, we design LE-Net Large, a scaled-up variant of LE-Net. Since the performance of modern DNNs is known to be highly dependent on their network depths and widths (He et al., 2016a; Tan and Le, 2019), we construct LE-Net Large by increasing the

TABLE A.2: Detailed comparison of LE-Net (Ours), LE-Net Large (Ours), and PointNet++. Class-wise precision (%), recall (%), and IoU (%) of each method are reported. Texts in parentheses indicate improvements over PointNet++.

		Collapsed	Non-Col.	BG
PointNet++	Pre.	58.6	89.8	96.0
	Rec.	64.8	92.9	93.3
	IoU	44.4	84.1	89.8
LE-Net (Ours)	Pre.	66.9 (+8.3)	91.2 (+1.4)	96.3 (+0.3)
	Rec.	67.8 (+3.0)	93.9 (+1.0)	94.4 (+1.1)
	IoU	50.8 (+6.4)	86.1 (+2.0)	91.1 (+1.3)
LE-Net Large (Ours)	Pre.	75.2 (+16.6)	92.4 (+2.6)	96.9 (+0.9)
	Rec.	74.6 (+9.8)	95.2 (+2.3)	95.0 (+1.7)
	IoU	59.9 (+15.5)	88.2 (+4.1)	92.2 (+2.4)

depth and width of LE-Net. Specifically, the input channel width C is increased from 128 to 150. Then, the depth of the network is increased by repeating LU-MLP blocks in each resolution level. Consequently, LE-Net Large is constructed as a 12-layer network (with two, two, six, and two LU-MLP blocks in the first, second, third, and fourth resolution levels, respectively).

As a result, as can be seen in Table A.1, LE-Net Large further advances upon LE-Net significantly at the expense of increased computational resources. Specifically, LE-Net Large achieves a 1.1% gain in OA and 4.1% gain in mIoU over LE-Net. Moreover, LE-Net Large outperforms the latest methods while requiring fewer computational resources. In particular, LE-Net Large outperforms the PointMLP, the second-best method, by 1.0% OA and 5.0% mIoU. Therefore, the proposed methods are effective in terms of overall performance.

With sufficient computational resources, LE-Net Large further improves detection performance significantly. See Table A.2. As with LE-Net, LE-Net Large significantly improves performance for the collapsed class, while providing relatively small improvements for other classes. In general, greater improvements compared to LE-Net are observed, demonstrating the full potential of LE-Net with increased computational resources.

A.2 Comparison of Different Local Operators

In this section, we compare different operators. Specifically, we replace the Δx_i in Equation 3.5 with different operators to compare performance. The result is shown in Table A.3.

Method	mIoU
Local Self-Attention	74.3
Gaussian	75.0
Average	75.4
Discrete Laplacian	76.0

TABLE A.3: Comparison of different local operators. mIoU (%) of each method are reported.

The first choice is renowned Self Attention (Vaswani et al., 2017). To adapt it to our setting, we implement the variant of Self Attention where its scope is limited to local neighbors. As can be seen, our choice (Discrete Laplacian) outperforms it significantly. Next, we use a gaussian weighted combination of point features to replace the discrete Laplacian. As a result, we observe that our choice outperforms it. Finally, we use a simple average filter to replace the discrete Laplacian. As shown in Table A.3, the discrete Laplacian outperforms it by 0.6.

A.3 Spectrum of building point clouds

In this study, we primarily use the discrete Laplacian for encoding edges or high-frequency components. In this section, we provide an analysis based on the Fourier transform, which can also be used to detect high-frequency components. Specifically, we create the digital surface model from point clouds and apply a 2D Fourier transform to obtain the spectrum of the digital surface model. A grid is applied to points and the maximum height within grid cells is taken as the value of the cell. Then, we apply frequency mask and inverse transform to analyze the distribution of high-frequency components. The result is shown in Figure A.1. As we can see from the spectrum, apart from the vertical and horizontal patterns, a non-collapsed building shows a regular diagonal pattern of frequencies. This is because the left and right sides of the roof edges are quite salient. As expected, high-frequency components appear on the edges of the roof. On the other hand, we find in the second row that many frequencies are occupied by a totally collapsed building. The shape of the building is quite irregular due to the collapse; thus, it requires many frequencies to reconstruct the image. Since a building becomes smoother when it collapses to the ground, the high-frequency component appears outside the building footprint. The third row shows an inclined collapsed building of which the roof is not so damaged. We observe that the frequencies used are relatively fewer compared with the

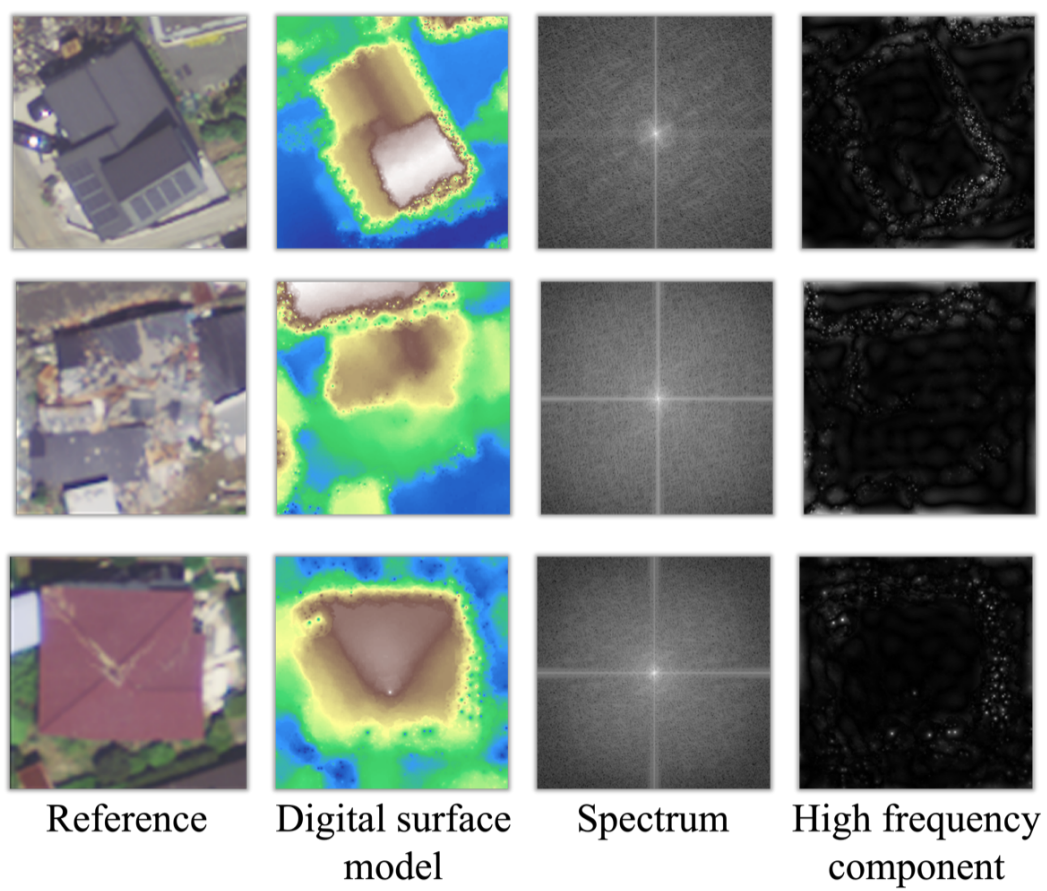


FIGURE A.1: Result of Fourier analysis.

totally collapsed one. The high-frequency components mainly appear on the edges that are still above the ground after the collapse.

In this study, we prefer the discrete Laplacian because, first, there is no trivial way to apply Fourier transform to point clouds directly. Second, the conversion to images/voxels causes information loss, and hence not desired.

A.4 Use of spatial geometry in the Laplacian Unit

In this section, we validate the usefulness of explicit inclusion of 3D geometry (positions) in the computation of LU. We perform the experiment using the Mashiki dataset. The result is shown below: As can be seen, LU w/ and

Method	mIoU
LU	76.0
LU with positions	75.9

TABLE A.4: Usefulness of 3D geometry in LU

w/o geometry achieves a similar performance. Therefore, we conclude that the inclusion of geometry is not necessary for LU computation.

Appendix B

Application of LU to Other Tasks

The content of this chapter is mainly based on the technical report (Xiu et al., 2022a).

B.1 Application to Shape Part Segmentation

Dataset We adopt widely used ShapeNet Part (Yi et al., 2016) dataset for part segmentation. This dataset contains 16,880 synthetic 3D objects. Categories included are some common objects like hats and knives. It contains a total of 16 object categories with 50 part categories where each object is annotated into two to six parts. Each point cloud contains around 2,300 points. For benchmarking purposes, we use the data provided by (Qi et al., 2017b). The standard train-test split in which 14,006 models are used for training and 2,874 models for testing is adopted.

Result As shown in the first row of Fig. B.1, with LUs, the network produces more precise predictions in near boundary regions compared to the plain counterpart. Therefore, LUs are especially effective in locating object boundaries. This point is also verified in the second row of Fig. B.1 in which the network manages to detect very subtle changes of the surface and prevents over-smoothing. In contrast to the plain network which is confused by similar parts (e.g., tabletop and drawer, the third row of Fig. B.1), the network clearly identifies two parts as separated ones and produces more perceptually sound predictions. We conjecture that the recognition of boundaries helps the network to separate different parts.

B.2 Application to Scene Segmentation

Dataset We use Stanford Large-Scale 3D Indoor Spaces (S3DIS) (Armeni et al., 2016) for scene segmentation. In total, six indoor environments (areas)

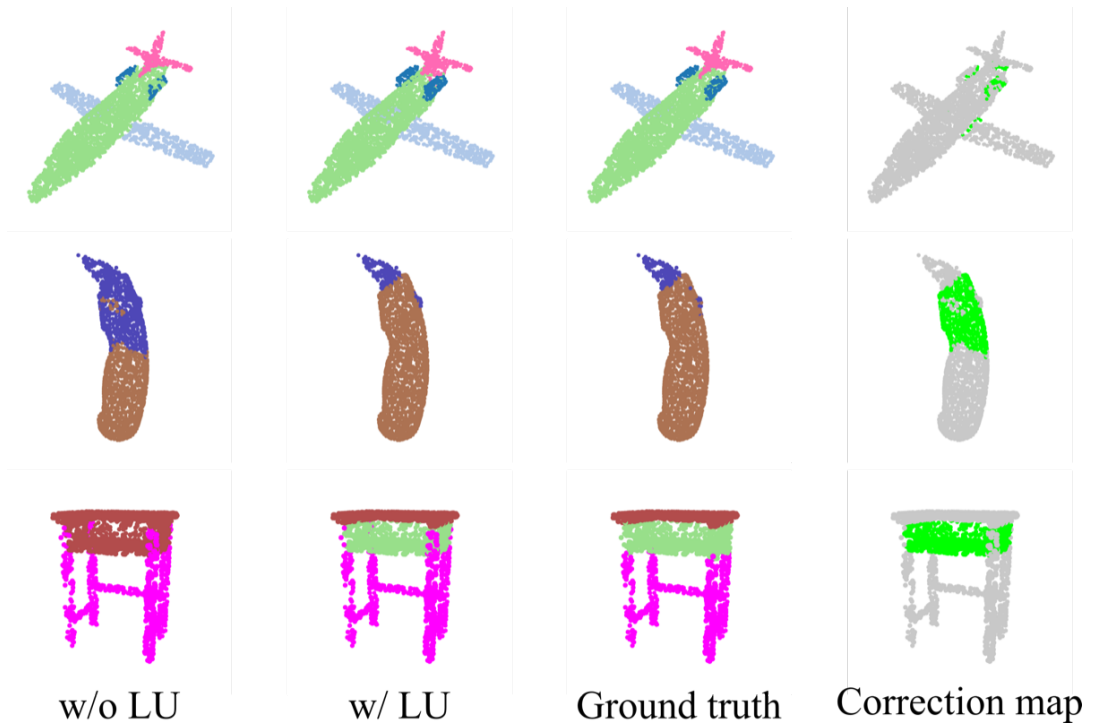


FIGURE B.1: Qualitative result of part segmentation. The correction map indicates the predictions that are corrected by combining LUs with the plain network. Green points show the corrected prediction. Compared with the plain network (w/o LU), the network with LU (w/ LU) makes the network more sensitive to part boundaries, which prevents over-smoothing and results in more perceptually sound predictions.

containing 272 rooms are included. Each point is labeled with a class from 13 categories. The number of points contained in a room ranges from 0.2M to 4.5M. As suggested by (Tchapmi et al., 2017), we use Area-5 for testing and others for training.

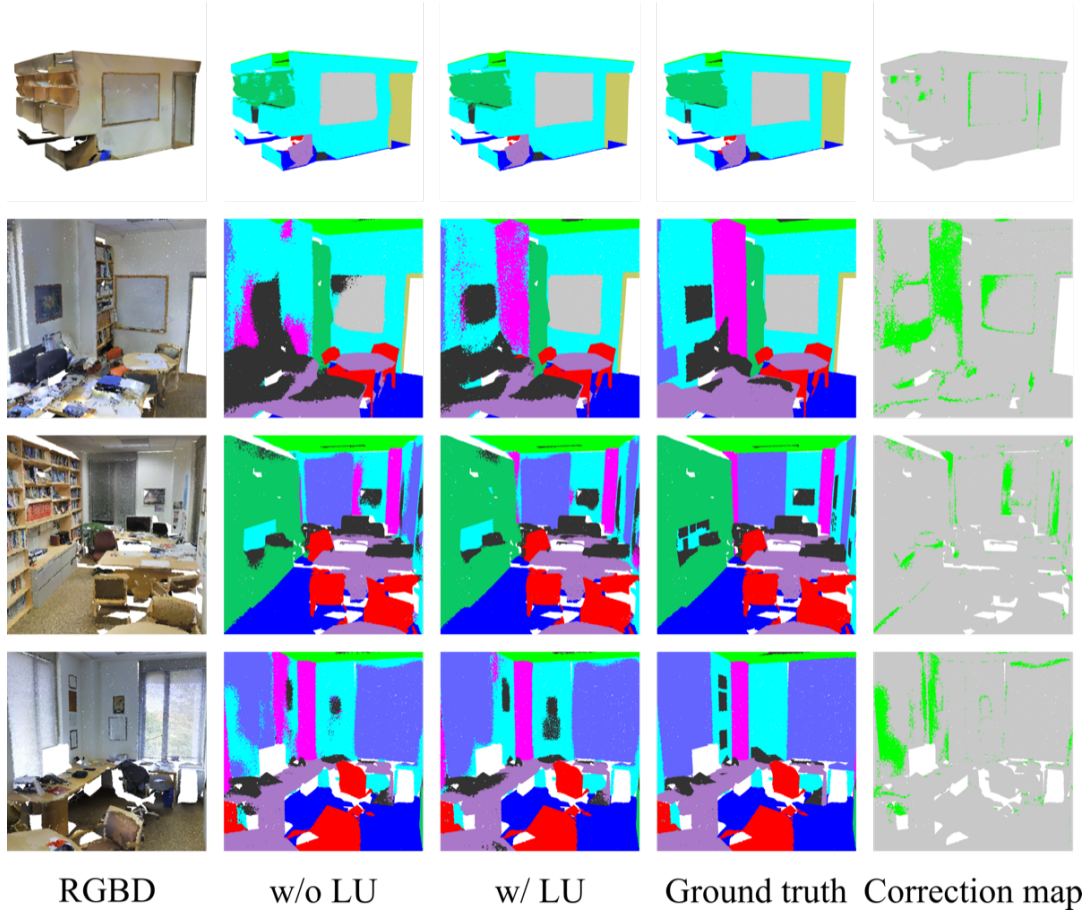


FIGURE B.2: Qualitative result of scene segmentation. The correction map indicates the predictions that are corrected by combining LUs with the plain network. **Green** points show the corrected prediction. Owing to its ability to perform smoothing and sharpening adaptively, LU encourages the network to produce sharp boundaries while keeping within-boundary regions smooth.

The primary effect of LU is the accurate localization of object boundaries. As can be seen in the correction map of the first row of Fig. B.2, the improvement is shown in near-boundary areas, making predictions more faithful to the ground truth. Secondly, owing to its ability to perform adaptive smoothing, the network with LUs, in general, produces smoother predictions. For instance, the fourth row of Fig. B.2 shows that not only the boundary of the window is recognized precisely but also the predictions of the within-boundary region present a smoother distribution compared to the one without LUs. Furthermore, with

the increased sensitivity to the geometrical/semantic changes of the surface, the network with LUs detects objects that are completely ignored by the plain counterpart and considerably improves the predictions both qualitatively and quantitatively (e.g., the second row of Fig. B.2). We believe that giving the freedom to perform smoothing or sharpening to LU makes the network more aware of the connectivity of the underlying surface, thus making predictions smoother within and sharper near object boundaries.

B.3 Curvature Analysis

To shed light on the underlying mechanism of LU, we can evaluate the change of the mean curvature before and after applying LU. Specifically, the increased curvature implies that the local surface undergoes a sharpening while the curvature becomes small when it is smoothed.

We provide qualitative analysis concerning the underlying mechanism of LU and explain how LU improves performance. As we have described, the behavior of LU can be investigated by analyzing the change of curvatures. Specifically, we measure the impact of LU by inspecting the change of curvatures qualitatively before and after applying LU. Three quantities are used for analysis: the input curvature $H_{in} = \|\sum_{j \in \mathcal{N}(i)} x_j - x_i\|$, the output curvature $H_{out} = \|\sum_{j \in \mathcal{N}(i)} x'_j - x'_i\|$, and $H_{\Delta x_i} = \|\Delta x_i\|$. Comparing H_{in} with H_{out} reveals the effect of LU on the smoothness of local surfaces. Furthermore, contrasting H_{in} and $H_{\Delta x_i}$ shows more directly how LU transforms the local features. We use the ShapeNet (part segmentation) and S3DIS (scene segmentation) for the analysis. The results are shown in Fig. B.3 and Fig. B.4, respectively.

As indicated by the yellow arrows in Fig. B.3 and B.4, LU performs smoothing in some cases just as indicated by its connection to mean curvature flow. As can be seen from the first row of Fig. B.4, for instance, small variations on the ceiling and the wall of the room are smoothed. Notice that LU does not blindly smooth out everything; in fact, we observe that the sharp edges of the objects remain salient while within-boundary regions are smoothed. Therefore, LU is able to perform smoothing selectively.

On the other hand, as can be seen from the second rows of Fig. B.3 and Fig. B.4, LU sharpens features by increasing their curvatures in some situations, which is in stark contrast to the mean curvature flow that only performs smoothing. More importantly, LU performs selective sharpening by increasing curvatures for some object edges whereas non-edge regions remain smooth. Furthermore, the fact that most sharpened edges correspond well to the ground

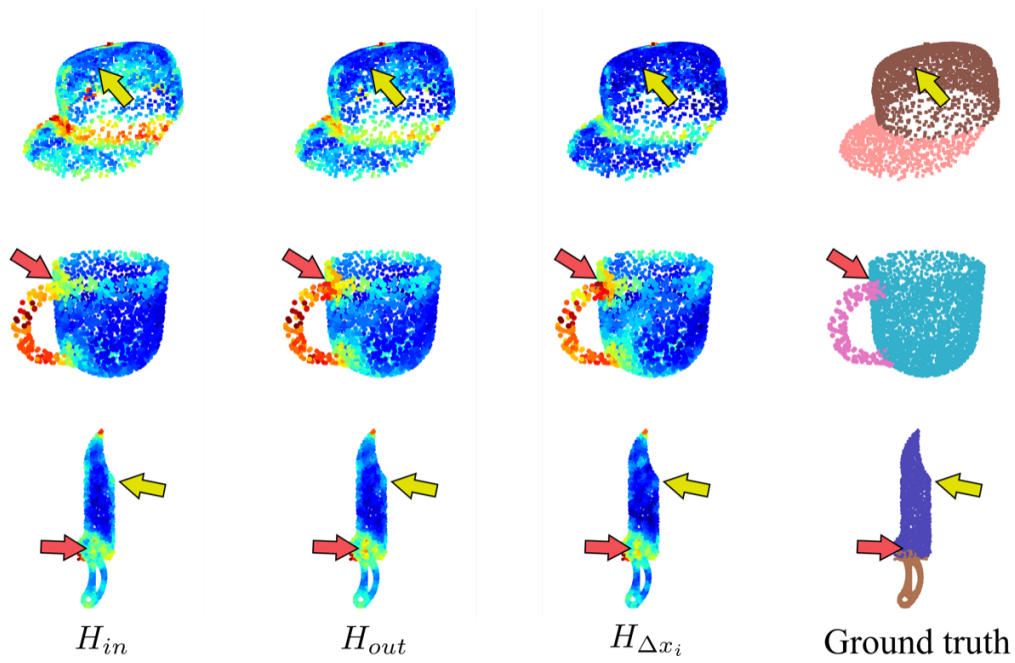


FIGURE B.3: Visualization of curvatures using the ShapeNet Part dataset. Yellow arrows indicate the region smoothed by LU whereas red arrows point to the sharpened one.

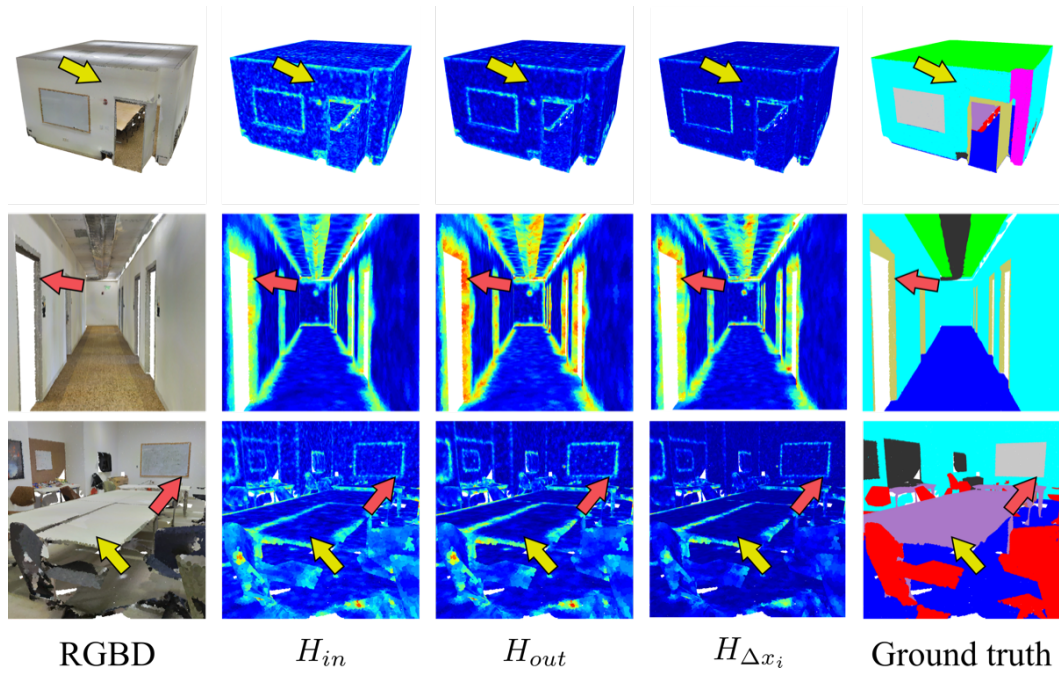


FIGURE B.4: Visualization of curvatures using the S3DIS dataset. Yellow arrows indicate the region smoothed by LU whereas red arrows point to the enhanced one.

truth object boundaries reveals that the effect of LU is task-dependent, further demonstrating its adaptability.

Apart from the cases where LU dominantly performs smoothing or sharpening, we observe that LU can simultaneously perform smoothing and sharpening to different parts of a single point cloud. Such situations are described by the last rows of Fig. B.3 and Fig. B.4. Scrutinizing H_{in} and H_{out} reveals that LU manages to remove small variations for intraregion points while enhancing points near object boundaries. Such an effect is more frequently observed in deeper layers where features are highly semantic.

Appendix C

Using Risai Damage Grade as Ground Truth

The content of this chapter is mainly based on Xiu et al. (2021b).

In this chapter, we use the label obtained from the field survey (Cabinet Office of Japan, 2018) directly as ground truth labels. It includes five classes: “No”, “Minor”, “Moderate-”, “Moderate+”, and “Major”. In addition, the “Story-collapsed” class is added by visually interpreting the “Major” buildings (Kawabe et al., 2021). The dataset is a subset of the Mashiki dataset. Machine learning methods as well as deep learning methods are compared.

C.1 Hand-Crafted Feature Calculation

Different types of commonly used features were used for supervised classification tasks. Most of the geometrical features were defined by following the work of Thomas et al. (2018). Besides, we defined a few additional features as follows: AGL (Above Ground Level) was computed using LAStools; Height difference features were computed as the difference between the elevation value of a selected point and the lowest point within its 15m radius neighborhood; Instead of color features, which are not available, point-wise backscattered amplitude information with its mean and variance in the defined neighborhood was computed; Return count information was obtained directly from LiDAR data; Horizontal angle feature that represents the angle between the local surface normal with the horizontal plane was calculated for all points within 1m neighborhood. All the features excluding AGL, Height difference, Amplitude, Return count information and Horizontal angle was repeatedly computed for multiple definitions of the neighborhood to capture multi-scale features. The scales were experimentally defined as 2m, 4m, and 8m, respectively. Therefore, 66 features were used for classification in total.

C.2 Algorithms

Several machine/deep learning algorithms are employed for classifications.

Random Forests Random Forest (Breiman, 2001) is a classifier that consists of multiple Decision Trees. Decision tree is a type of method that recursively splits the feature space into a set of rectangles and then fits a value (in the case of regression) or a majority vote (in the case of classification). An individual tree is a conceptually simple yet powerful model that can approximate highly non-linear relationships of data by choosing the input features and split-point (feature value). In the case of classification, Random Forests takes a majority vote from all individual trees to generate the final vote. Improving the performance of a machine learning algorithm is essentially a trade-off between model variance reduction and bias reduction. Novelty of Random Forests is to improve the variance reduction by reducing the correlation between individual trees in the tree-growing process. Specifically, this de-correlation is achieved by a random feature sub-sample when deciding a split point. For a further theoretical explanation of the algorithm, please refer to Breiman (2001).

LightGBM LightGBM (Ke et al., 2017) is an algorithm developed based on the idea of boosting. Similar to Random Forests, the committee of classifiers is involved in boosting. However, instead of getting a majority vote as classification results, boosting sequentially applies classifiers to the modified version of data. Each data point in the dataset fit by a classifier is individually weighted, and these weights are updated during the training process. The main idea of boosting is to sequentially apply classifiers to correct the errors made by previous classifiers and thereby produce an overall good model at last by weighted majority vote. Based on the fundamental boosting principle, LightGBM additionally implements Gradient-based One-Side Sampling and Exclusive Feature Bundling to improve the efficiency and scalability of the model. In this study, we adopted LightGBM with Gradient Boosting Decision Trees for classification tasks.

PointNet PointNet (Qi et al., 2017a) is a deep neural network-based algorithm that directly takes raw 3D point cloud as input. PointNet is an innovative deep neural network that firstly consumes raw point cloud and obtains point-wise features by shared multi-layer perceptrons followed by max-pooling to aggregate individual features to generate global features. Then the classification is done by stacking local and global features together, and hence both individual point, as well as their neighborhood information, is considered. This simple and innovative idea achieved state-of-the-art performance at the time of

publication and it is still a competitive algorithm for various tasks.

DSGCN Different from PointNet, DSGCN (Dynamic-Scale Graph Convolutional Network) proposed by Xiu, Shinohara, and Matsuoka (2019) is a deep neural network that takes a point cloud in the form of dynamically changing k-neighbor graph in order to model the complex local neighborhood structures. The k-neighbor graphs are dilated or shrunk randomly at each sampling iteration to capture scale-invariant features in a computationally efficient manner. Furthermore, different scales of the k-neighbor graphs are combined to generate multi-scale features for final predictions. The experimental results showed that DSGCN is accurate while being robust to data corruption.

C.3 Results and Discussion

In this section, experimental results with five-fold cross-validation are illustrated and explained. In total, three types of experiments were conducted: “Major and Story-collapsed” and “Other damage” classification, “Story-collapsed” and “Other damage” classification, and separated damage classification. Note that “No” damage buildings were combined with “Minor” damage buildings because they have a too small number of samples.

C.3.1 “Background”, “Major and Story-collapsed” and “Other damage” classification

It is of paramount importance to recognize higher-grade building damages to rescue people in danger as soon as possible. According to Figure C.1, the “Background” and “Other damage” classes are well separated while “Major and Story-collapsed” classes scored lower than them. The “Background” class is well separated from other classes because a majority of them are simple objects such as ground and trees. The main misclassifications were introduced mainly “Major and Story-collapsed” class. “Major” class contains not only the “crashed” buildings but also the buildings with greater inclination or having damage at their base but without visible exterior deformation. For the former case, the characteristics of the building are closer to the “Story-collapsed” class while it is closer to the “Other damage” class for the latter case. On the other hand, the major ambiguity between “Background” and “Major and Story-collapsed” is debris. The classifiers were able to detect debris but increased false alarms such as low trees near the houses and ground with large slopes. Compared to

Random Forests and LightGBM, deep learning-based models have lower overall performance and increased variance for all classes. This result may be due to the small sample population as deep learning usually needs more samples than traditional machine learning methods. Another presumable reason is the imbalanced distribution of training data. Unlike Random Forests and LightGBM, which received the same number of training samples for each class, the training samples were drawn randomly in every sampling iteration. Therefore, the sampled training data should approximately follow the original distribution, which is extremely imbalanced. Under such a condition, DSGCN outperformed PointNet in every metric. The increase in performance probably attributes to the exploitation of local structures by the k-neighbor graph, which can model the interactions among neighboring points better than PointNet.

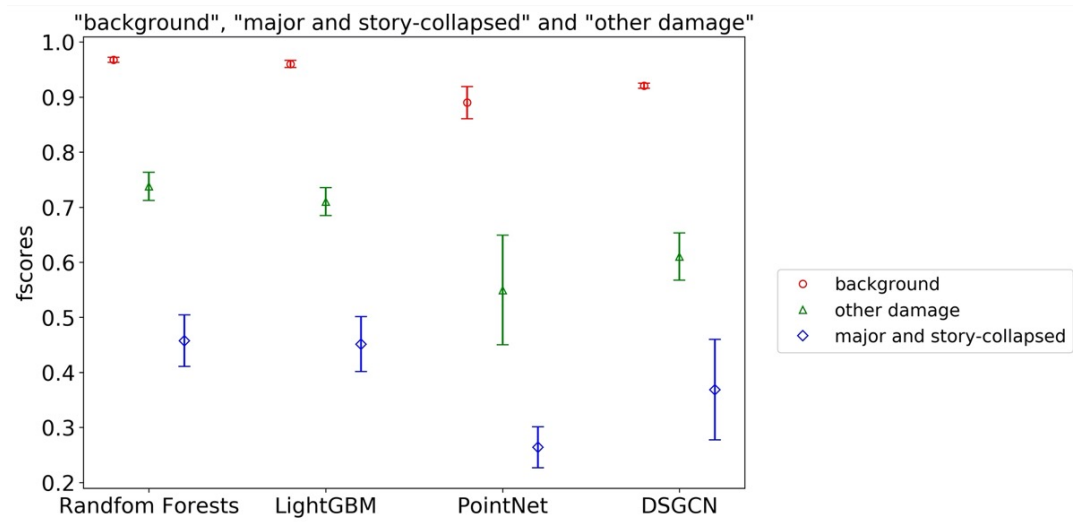


FIGURE C.1: Classification results of “Background”, “Major and Story-collapsed” and “Other damage”

C.3.2 “Background”, “Story-collapsed” and “Other damage” classification

To justify the arguments made in the former experiments, in this experiment we combined “major damage” into the “Other damage” class. As a result, Figure C.2 illustrates that the misclassification of “Other damage” decreased and thus the performance was vastly increased by approximately 0.2 in terms of f-score. This result is consistent with our initial arguments. However, the ambiguity between the “crashed” “Major” class and the “Story-collapsed” class still exists so that the ground and debris are inseparable. The relatively high variance of the “Story-collapsed” class is possibly due to the reduction of available training samples

since the “Major” class was moved to the “Other damage” class. This is most obvious for deep learning-based models, which scored poorly because of reduced samples.

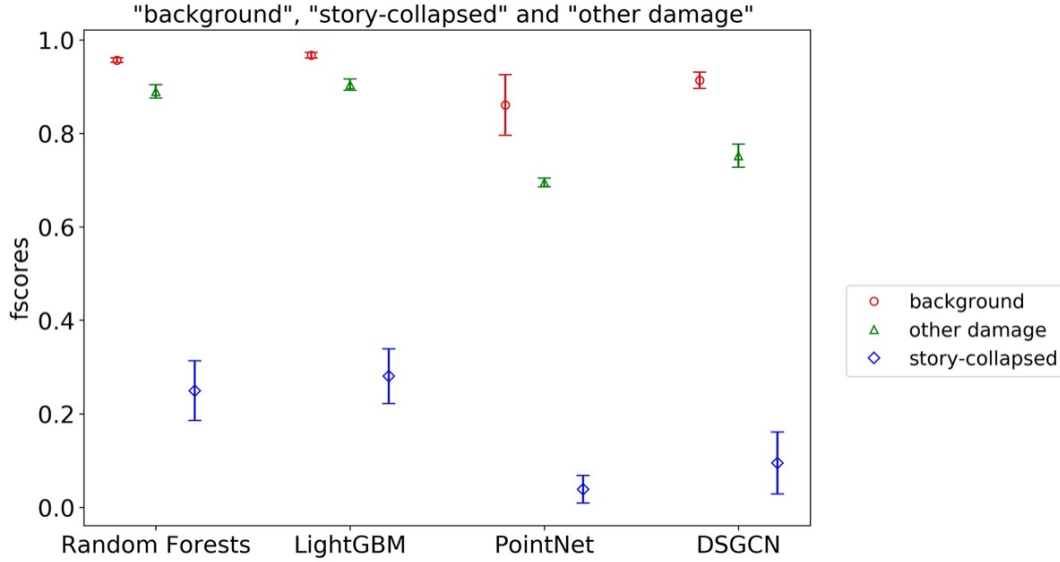


FIGURE C.2: Classification results of “Background”, “Story-collapsed” and “Other damage”

C.3.3 Separated damage classification

To investigate the separability of each damaged building class, we have implemented six class classification experiments. The overall results shown in Figure C.3 were consistent with our intuition that extreme cases such as “Major” and “Story-collapsed” are easier to be identified, while intermediate classes are difficult to be correctly identified. The misclassification among intermediate classes was proportional to the sample numbers, which indicated that the classes are almost inseparable given used hand-crafted features. Despite deep learning-based methods did not rely on hand-crafted features, they suffered from insufficient sample populations, and hence their performances were not stable. Furthermore, it is most obvious in this classification result that multiple damage grade predictions exist in one building, which is impossible in reality.

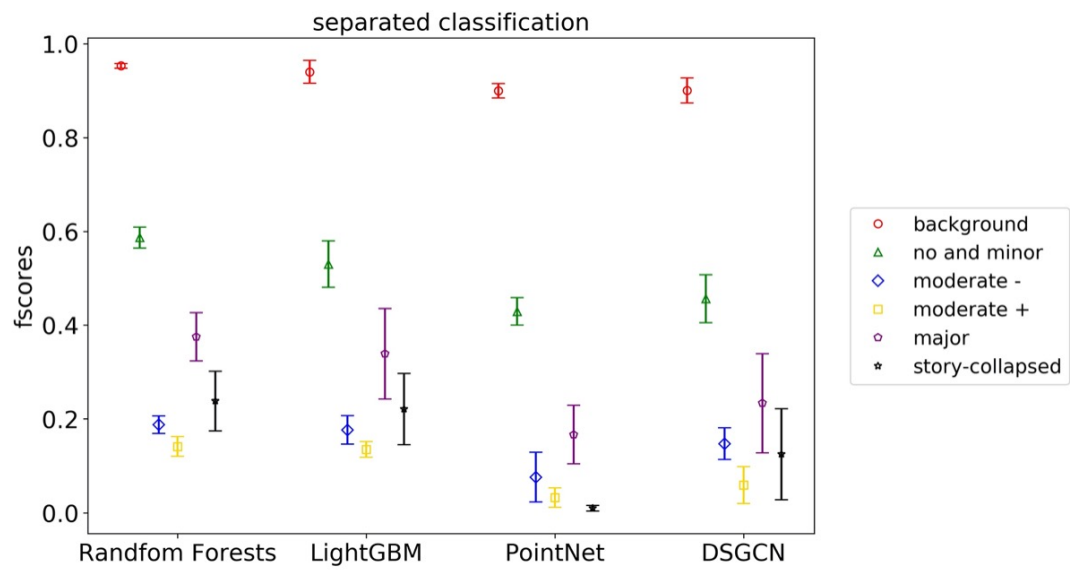


FIGURE C.3: Results of separated damage classification

Appendix D

Publications Related to This Research

Publications related to Chapter 2 are as follows:

- Preliminary analysis of the dataset (Xiu et al., 2021b). International conference.
- Dataset development and benchmarking (Xiu et al., 2020). Peer-reviewed journal.
- Statistical analysis of the dataset (Xiu et al., 2022b). Peer-reviewed journal,

Publications related to Chapter 3 and Chapter 4 are as follows:

- Development of LU and LE-Net (Xiu et al., 2023). Peer-reviewed journal.
- A technical report of LU (Xiu et al., 2022a). Preprint.

Appendix E

Detailed explanations of the discrete Laplacian

In this chapter, we explain in detail the discrete Laplacian which is the main tool we exploit to build LE-Net.

E.1 Finite difference approximation of 1D Laplacian

In this section, we explain the finite difference approximation of the Laplacian. For simplicity, we assume the function is 1D throughout this section. The Laplacian is defined as the divergence of the gradient of the function f . Mathematically, the Laplacian can be expressed as:

$$\Delta f = \nabla \cdot \nabla f = \frac{\partial^2 f}{\partial x^2} = f''(x) \quad (\text{E.1})$$

For implementation, we need to discretize the above equation to compute it using computers. Although there are multiple ways to discretize the Laplacian, we use the finite difference method to discretize the Laplacian for its simplicity. Specifically, we first approximate the first-order derivative of f using the central difference:

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h}, \quad (\text{E.2})$$

where h is a step size. Using the above formula, we can obtain the expression for $f'(x + \frac{h}{2})$ and $f'(x - \frac{h}{2})$:

$$f'(x + \frac{h}{2}) = \frac{f(x+h) - f(x)}{h}. \quad (\text{E.3})$$

$$f'(x - \frac{h}{2}) = \frac{f(x) - f(x-h)}{h} \quad (\text{E.4})$$

By combining the Equation (E.2), (E.3), and (E.4), we obtain the finite difference approximation of the second-order derivative:

$$f''(x) = \Delta f = \frac{f(x+h) + f(x-h) - 2f(x)}{h^2}, \quad (\text{E.5})$$

which equals the Laplacian. The key observation is that the Laplacian first adds the neighbor values and then subtracts the center values by a number equal to the number of neighbors. In other words, the Laplacian aggregates the difference between neighbors and the center, thus revealing how different the center point from neighbors is.

E.2 Image Laplacian to the point Laplacian

In the case of a 2D image $I(x, y)$ with one channel (e.g., intensity), the Laplacian is defined as:

$$\Delta I = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}. \quad (\text{E.6})$$

Similar to the 1D case, the above equation is discretized as below:

$$\Delta f = f(x+1, y) + f(x, y+1) + f(x-1, y) + f(x, y-1) - 4f(x, y), \quad (\text{E.7})$$

where h is assumed to be 1, a step in a 2D grid.

Alternatively, we can define a vector field $\mathbf{X} : \mathbb{R}^2 \rightarrow \mathbb{R}^3$, which has the explicit form:

$$\mathbf{X}(x, y) = (x, y, I(x, y)). \quad (\text{E.8})$$

In doing so, we model the image as a parametric surface parameterized by coordinates x and y . For this vector field, we can compute its (discretized) Laplacian as:

$$\Delta \mathbf{X}(x, y) = (\Delta x, \Delta y, \Delta I) = (0, 0, \Delta I). \quad (\text{E.9})$$

As we can see, the first and second components of Equation (E.9) are always zero for images because the neighbors are symmetrically placed with the same distance from the center. Next, consider a case where those pixels are replaced with points. In this case, the Laplacian still is the same as Equation (E.9) because points are still equally and symmetrically spaced. However, the situation changes if neighbor points are *not* equally spaced. Such a case is shown in Figure E.1. As can be seen, the first and second components of the Laplacian are no longer zero because points are irregularly distributed. The use of such information intuitively makes sense because non-zero components contain some

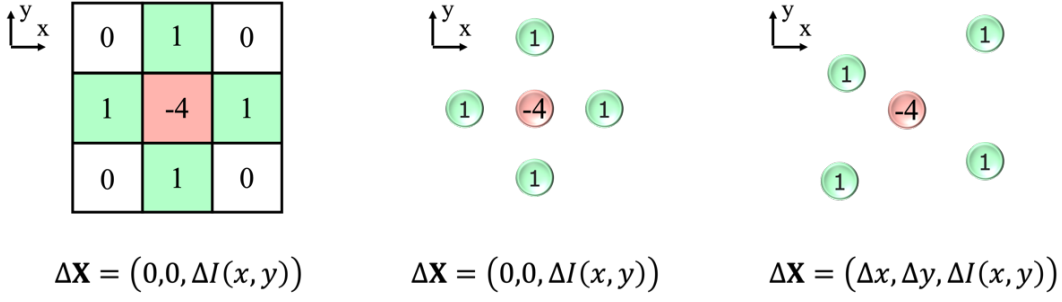


FIGURE E.1: Comparison of discrete image and point Laplacians. From left to right: image (2D grid) Laplacian, equidistant point Laplacian, and Laplacian for 3D point clouds.

useful information about the local geometry. More generally, unlike images, those weights on the points can be non-zero since they are not equidistant.

We can straightforwardly extend this formulation to 3D point clouds with some attributes (e.g., intensity). Let $P(x, y) = z$ be the point cloud that outputs the height. We can similarly define a vector field:

$$\mathbf{X}(x, y) = (x, y, P(x, y) = z), \quad (\text{E.10})$$

where z represents the height. Its Laplacian can also be computed as:

$$\Delta \mathbf{X}(x, y) = (\Delta x, \Delta y, \Delta P). \quad (\text{E.11})$$

Now let $\mathbf{x}_i = \mathbf{X}(x_i, y_i, z_i)$ be the feature of point i and j denote the neighbors within a certain neighborhood $\mathcal{N}(i)$ of point i . The generalized discrete Laplacian used in this study is defined as:

$$\Delta \mathbf{x}_i = \sum_{j \in \mathcal{N}(i)} w_{ij} \cdot (\mathbf{x}_j - \mathbf{x}_i), \quad (\text{E.12})$$

where w_{ij} is the weight assigned to a particular center-neighbor pair and satisfies $\sum_{j \in \mathcal{N}(i)} w_{ij} = 1$. Although there are many choices for w_{ij} , we adopt $\frac{1}{\mathcal{N}(i)}$ for its simplicity in this study.

Bibliography

- Alzubaidi, Laith et al. (2021). “Review of deep learning: Concepts, CNN architectures, challenges, applications, future directions”. In: *Journal of big Data* 8.1, pp. 1–74.
- Armeni, Iro et al. (2016). “3d semantic parsing of large-scale indoor spaces”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1534–1543.
- Bai, Yanbing et al. (2017). “A framework of rapid regional tsunami damage recognition from post-event TerraSAR-X imagery using deep neural networks”. In: *IEEE Geoscience and Remote Sensing Letters* 15.1, pp. 43–47.
- Breiman, Leo (2001). “Random forests”. In: *Machine learning* 45.1, pp. 5–32.
- Cabinet Office of Japan (2018). *Operational Guideline for Damage Assessment of Residential Buildings in Disasters*. http://www.bousai.go.jp/taisaku/pdf/h3003shishin_all.pdf. Accessed: 2022-10-28.
- Castelluccio, Marco et al. (2015). “Land Use Classification in Remote Sensing Images by Convolutional Neural Networks”. In: *CoRR* abs/1508.00092. arXiv: 1508.00092. URL: <http://arxiv.org/abs/1508.00092>.
- Chen, Hongming et al. (2018). “The rise of deep learning in drug discovery”. In: *Drug discovery today* 23.6, pp. 1241–1250.
- Cheng, Min-Lung, Toshiaki Satoh, and Masashi Matsuoka (2018). “Region-Based Co-Seismic Ground Displacement Detection Using Optical Aerial Imagery”. In: *IGARSS 2018-2018 IEEE International Geoscience and Remote Sensing Symposium*. IEEE, pp. 2940–2943.
- Chiba, Tatsuro (n.d.). *Post-Kumamoto Earthquake (16 April 2016) Rupture Lidar Scan*. Air Asia Survey Co., Ltd, distributed by OpenTopography. <https://portal.opentopography.org/datasetMetadata?otCollectionID=OT.052018.2444.1>. DOI: 10.5069/G9SX6B9T.
- Chollet, François (2017). “Xception: Deep learning with depthwise separable convolutions”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1251–1258.
- Ci, Tianyu, Zhen Liu, and Ying Wang (2019). “Assessment of the degree of building damage caused by disaster using convolutional neural networks in combination with ordinal regression”. In: *Remote Sensing* 11.23, p. 2858.

- Cooner, Austin J, Yang Shao, and James B Campbell (2016). “Detection of urban damage using remote sensing and machine learning algorithms: Revisiting the 2010 Haiti earthquake”. In: *Remote Sensing* 8.10, p. 868.
- Deng, Jia et al. (2009). “Imagenet: A large-scale hierarchical image database”. In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee, pp. 248–255.
- Desbrun, Mathieu et al. (1999). “Implicit fairing of irregular meshes using diffusion and curvature flow”. In: *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pp. 317–324.
- Dong, Laigen and Jie Shan (2013). “A comprehensive review of earthquake-induced building damage detection with remote sensing techniques”. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 84, pp. 85–99.
- Dong, Xiaowen et al. (2016). “Learning Laplacian matrix in smooth graph signal representations”. In: *IEEE Transactions on Signal Processing* 64.23, pp. 6160–6173.
- Eberhard, Marc O et al. (2010). “The mw 7.0 Haiti earthquake of January 12, 2010: USGS/EERI advance reconnaissance team report”. In: *US Geological Survey Open-File Report* 1048.2013, p. 64.
- Elberink, S Oude et al. (2011). “Detection of collapsed buildings by classifying segmented airborne laser scanner data”. In: *Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci* 38.5, W12.
- Farabet, Clement et al. (2012). “Learning hierarchical features for scene labeling”. In: *IEEE transactions on pattern analysis and machine intelligence* 35.8, pp. 1915–1929.
- Fernandez Galarreta, J, N Kerle, and M Gerke (2015). “UAV-based urban structural damage assessment using object-based image analysis and semantic reasoning”. In: *Natural hazards and earth system sciences* 15.6, pp. 1087–1101.
- Gamba, Paolo and Fabio Casciati (1998). “GIS and image understanding for near-real-time earthquake damage assessment”. In: *Photogrammetric engineering and remote sensing* 64, pp. 987–994.
- Gerke, Markus and Norman Kerle (2011). “Automatic structural seismic damage assessment with airborne oblique Pictometry© imagery”. In: *Photogrammetric Engineering & Remote Sensing* 77.9, pp. 885–898.
- Glorot, Xavier, Antoine Bordes, and Yoshua Bengio (2011). “Deep sparse rectifier neural networks”. In: *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, pp. 315–323.

- Goodfellow, Ian, Yoshua Bengio, and Aaron Courville (2016). *Deep learning*. MIT press.
- Grünthal, Gottfried (1998). *European macroseismic scale 1998*. Tech. rep. European Seismological Commission (ESC).
- Gupta, Ritwik et al. (June 2019). “Creating xBD: A Dataset for Assessing Building Damage from Satellite Imagery”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*.
- Gupta, Rohit and Mubarak Shah (2020). “RescueNet: Joint Building Segmentation and Damage Assessment from Satellite Imagery”. In: *25th International Conference on Pattern Recognition, ICPR 2020, Virtual Event / Milan, Italy, January 10-15, 2021*. IEEE, pp. 4405–4411. DOI: [10.1109/ICPR48806.2021.9412295](https://doi.org/10.1109/ICPR48806.2021.9412295). URL: <https://doi.org/10.1109/ICPR48806.2021.9412295>.
- Hamaguchi, Ryuhei, Ken Sakurada, and Ryosuke Nakamura (2019). “Rare event detection using disentangled representation learning”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9327–9335.
- Hamdi, Abdullah, Silvio Giancola, and Bernard Ghanem (2021). “Mvtn: Multi-view transformation network for 3d shape recognition”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1–11.
- He, Kaiming et al. (2016a). “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778.
- He, Meizhang et al. (2016b). “A 3D shape descriptor based on contour clusters for damaged roof detection using airborne LiDAR point clouds”. In: *Remote Sensing* 8.3, p. 189.
- Hinton, Geoffrey et al. (2012a). “Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups”. In: *IEEE Signal processing magazine* 29.6, pp. 82–97.
- Hinton, Geoffrey E et al. (2012b). “Improving neural networks by preventing co-adaptation of feature detectors”. In: *arXiv preprint arXiv:1207.0580*.
- Hu, Jie, Li Shen, and Gang Sun (2018). “Squeeze-and-excitation networks”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7132–7141.
- Ioffe, Sergey and Christian Szegedy (2015). “Batch normalization: Accelerating deep network training by reducing internal covariate shift”. In: *International conference on machine learning*. PMLR, pp. 448–456.

- Jiang, Junguang et al. (2022). “Transferability in Deep Learning: A Survey”. In: *arXiv preprint arXiv:2201.05867*.
- JTC ROOFING CONTRACTORS LTD (n.d.). *28 TYPES OF ROOF DESIGNS & STYLES*. <https://jtcroofing.co.uk/roof-types/>. Accessed: 2023-02-10.
- Kalofolias, Vassilis (2016). “How to learn a graph from smooth signals”. In: *Artificial Intelligence and Statistics*. PMLR, pp. 920–929.
- Kamagatani, Y and M Matsuoka (2017). “Damaged building recognition of the 2016 Kumamoto earthquakes using deep learning with aerial photographs”. In: *Tono Research Institute of Earthquake Science “the report of Disaster Prevention Research Committee*, pp. 49–57.
- Kawabe, Ken et al. (2021). “Extraction of story-collapsed buildings by the 2016 Kumamoto Earthquakes using deep learning”. In: *17th World Conference on Earthquake Engineering*.
- Ke, Guolin et al. (2017). “Lightgbm: A highly efficient gradient boosting decision tree”. In: *Advances in neural information processing systems* 30.
- Khoshelham, Kourosh, Sander Oude Elberink, and Sudan Xu (2013). “Segment-Based Classification of Damaged Building Roofs in Aerial Laser Scanning Data”. In: *IEEE Geoscience and Remote Sensing Letters* 10.5, pp. 1258–1262. DOI: [10.1109/LGRS.2013.2257676](https://doi.org/10.1109/LGRS.2013.2257676).
- Khoshelham, Kourosh and SJ Oude Elberink (2012). “Role of dimensionality reduction in segment-based classification of damaged building roofs in airborne laser scanning data”. In: *Proceedings of the International Conference on Geographic Object Based Image Analysis, Rio de Janeiro, Brazil*, pp. 7–9.
- Kohiyama, Masayuki and Fumio Yamazaki (2005). “Damage detection for 2003 Bam, Iran, earthquake using Terra-ASTER satellite imagery”. In: *Earthquake Spectra* 21.1_suppl, pp. 267–274.
- Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton (2012). “Imagenet classification with deep convolutional neural networks”. In: *Advances in neural information processing systems*, pp. 1097–1105.
- Labiak, Richard C et al. (2011). “Automated method for detection and quantification of building damage and debris using post-disaster LiDAR data”. In: *Laser Radar Technology and Applications XVI*. Vol. 8037. SPIE, pp. 145–152.
- Lai, Xin et al. (2022). “Stratified Transformer for 3D Point Cloud Segmentation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8500–8509.

- Lan, Shiyi et al. (2019). “Modeling local geometric structure of 3d point clouds using geo-cnn”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 998–1008.
- Le, Truc and Ye Duan (2018). “Pointgrid: A deep network for 3d shape understanding”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 9204–9214.
- LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton (2015). “Deep learning”. In: *nature* 521.7553, pp. 436–444.
- Li, Yangyan et al. (2018). “Pointcnn: Convolution on x-transformed points”. In: *Advances in neural information processing systems*, pp. 820–830.
- Liu, Sikang et al. (2022). “Efficient SfM for Large-Scale UAV Images Based on Graph-Indexed BoW and Parallel-Constructed BA Optimization”. In: *Remote Sensing* 14.21, p. 5619.
- Liu, Yongcheng et al. (2019). “Relation-shape convolutional neural network for point cloud analysis”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8895–8904.
- Liu, Ze et al. (2020). “A closer look at local aggregation operators in point cloud analysis”. In: *European Conference on Computer Vision*. Springer, pp. 326–342.
- Ma, Xu et al. (2022). “Rethinking Network Design and Local Geometry in Point Cloud: A Simple Residual MLP Framework”. In: *arXiv preprint arXiv:2202.07123*.
- Maggiori, Emmanuel et al. (2016). “Convolutional neural networks for large-scale remote-sensing image classification”. In: *IEEE Transactions on geoscience and remote sensing* 55.2, pp. 645–657.
- Matsuoka, Masashi and Fumio Yamazaki (2004). “Use of satellite SAR intensity imagery for detecting building areas damaged due to earthquakes”. In: *Earthquake Spectra* 20.3, pp. 975–994.
- (2005). “Building damage mapping of the 2003 Bam, Iran, earthquake using Envisat/ASAR intensity imagery”. In: *Earthquake Spectra* 21.1_suppl, pp. 285–294.
- Maturana, Daniel and Sebastian Scherer (2015). “Voxnet: A 3d convolutional neural network for real-time object recognition”. In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 922–928.
- Miura, Hiroyuki, Tomohiro Aridome, and Masashi Matsuoka (2020). “Deep Learning-Based Identification of Collapsed, Non-Collapsed and Blue Tarp-Covered Buildings from Post-Disaster Aerial Images”. In: *Remote Sensing* 12.12, p. 1924.

- Miura, Hiroyuki, Saburoh Modorikawa, and S Hui Chen (2011). “Texture characteristics of high-resolution satellite images in damaged areas of the 2010 Haiti earthquake”. In: *Proceedings of the 9th International Workshop on Remote Sensing for Disaster Response, Stanford, CA, USA*, pp. 15–16.
- Montavon, Grégoire, Wojciech Samek, and Klaus-Robert Müller (2018). “Methods for interpreting and understanding deep neural networks”. In: *Digital Signal Processing* 73, pp. 1–15.
- Moya, Luis et al. (2018). “Detection of collapsed buildings from lidar data due to the 2016 Kumamoto earthquake in Japan”. In: *Natural Hazards and Earth System Sciences* 18.1, pp. 65–78.
- Nair, Vinod and Geoffrey E Hinton (2010). “Rectified linear units improve restricted boltzmann machines”. In: *Icml*.
- Naito, Shohei et al. (2020). “Building-damage detection method based on machine learning utilizing aerial photographs of the Kumamoto earthquake”. In: *Earthquake Spectra* 36.3, pp. 1166–1187.
- Nevo, Sella et al. (2019). “ML for flood forecasting at scale”. In: *arXiv preprint arXiv:1901.09583*.
- Okada, Shigeyuki and Nobuo Takai (2000). “Classifications of structural types and damage patterns of buildings for earthquake field investigation”. In: *Proceedings of the 12th world conference on earthquake engineering (paper 0705), Auckland*.
- Qi, Charles R et al. (2017a). “Pointnet: Deep learning on point sets for 3d classification and segmentation”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 652–660.
- Qi, Charles Ruizhongtai et al. (2017b). “Pointnet++: Deep hierarchical feature learning on point sets in a metric space”. In: *Advances in neural information processing systems*, pp. 5099–5108.
- Qian, Guocheng et al. (2022). “PointNeXt: Revisiting PointNet++ with Improved Training and Scaling Strategies”. In: *arXiv preprint arXiv:2206.04670*.
- Qiu, Shi, Saeed Anwar, and Nick Barnes (2021). “Geometric back-projection network for point cloud classification”. In: *IEEE Transactions on Multimedia*.
- Rehor, Miriam and Hans-Peter Bähr (2006). “Segmentation of Damaged Buildings from Laser Scanning Data”. In: *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences. Bonn, Germany:[sn], XXXVI, Part 3*, pp. 67–72.

- Rehor, Miriam et al. (2008). “Contribution of two plane detection algorithms to recognition of intact and damaged buildings in lidar data”. In: *The Photogrammetric Record* 23.124, pp. 441–456.
- Riegler, Gernot, Ali Osman Ulusoy, and Andreas Geiger (2017). “Octnet: Learning deep 3d representations at high resolutions”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3577–3586.
- Ronneberger, Olaf, Philipp Fischer, and Thomas Brox (2015). “U-net: Convolutional networks for biomedical image segmentation”. In: *International Conference on Medical image computing and computer-assisted intervention*. Springer, pp. 234–241.
- Rumelhart, David E, Geoffrey E Hinton, and Ronald J Williams (1986). “Learning representations by back-propagating errors”. In: *nature* 323.6088, pp. 533–536.
- Saito, Keiko et al. (2004). “Using high-resolution satellite images for post-earthquake building damage assessment: a study following the 26 January 2001 Gujarat earthquake”. In: *Earthquake spectra* 20.1, pp. 145–169.
- Sakamoto, Mitsuteru et al. (2004). “Automatic detection of damaged area of Iran earthquake by high-resolution satellite imagery”. In: *IGARSS 2004. 2004 IEEE International Geoscience and Remote Sensing Symposium*. Vol. 2. IEEE, pp. 1418–1421.
- Schweier, Christine and Michael Markus (2006). “Classification of collapsed buildings for fast damage and loss assessment”. In: *Bulletin of earthquake engineering* 4.2, pp. 177–192.
- Selvaraju, Ramprasaath R et al. (2017). “Grad-cam: Visual explanations from deep networks via gradient-based localization”. In: *Proceedings of the IEEE international conference on computer vision*, pp. 618–626.
- Shen, Yu et al. (2021). “Bdanet: Multiscale convolutional neural network with cross-directional attention for building damage assessment from satellite images”. In: *IEEE Transactions on Geoscience and Remote Sensing* 60, pp. 1–14.
- Simonyan, Karen and Andrew Zisserman (2014). “Very deep convolutional networks for large-scale image recognition”. In: *arXiv preprint arXiv:1409.1556*.
- Song, Hwanjun et al. (2022). “Learning from noisy labels with deep neural networks: A survey”. In: *IEEE Transactions on Neural Networks and Learning Systems*.
- Sorkine, Olga (2006). “Differential representations for mesh processing”. In: *Computer Graphics Forum*. Vol. 25. 4. Wiley Online Library, pp. 789–807.

- Su, Hang et al. (2015). “Multi-view convolutional neural networks for 3d shape recognition”. In: *Proceedings of the IEEE international conference on computer vision*, pp. 945–953.
- Sugiyama, MITGT and Hitoshi Saji Keiichi Abe (2002). “Detection of earthquake damaged areas from aerial photographs by using color and edge information”. In: *Proceedings of the 5th Asian Conference on Computer Vision, Melbourne, Australia*, pp. 23–25.
- Szegedy, Christian et al. (2015). “Going deeper with convolutions”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9.
- Szegedy, Christian et al. (2017). “Inception-v4, inception-resnet and the impact of residual connections on learning”. In: *Thirty-first AAAI conference on artificial intelligence*.
- Tan, Mingxing and Quoc Le (2019). “Efficientnet: Rethinking model scaling for convolutional neural networks”. In: *International conference on machine learning*. PMLR, pp. 6105–6114.
- Taubin, Gabriel (1995). “A signal processing approach to fair surface design”. In: *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pp. 351–358.
- Tchapmi, Lyne et al. (2017). “Segcloud: Semantic segmentation of 3d point clouds”. In: *2017 international conference on 3D vision (3DV)*. IEEE, pp. 537–547.
- Thomas, Hugues et al. (2018). “Semantic classification of 3D point clouds with multiscale spherical neighborhoods”. In: *2018 International conference on 3D vision (3DV)*. IEEE, pp. 390–398.
- Thomas, Hugues et al. (2019). “Kpconv: Flexible and deformable convolution for point clouds”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6411–6420.
- Tong, Xiaohua et al. (2012). “Building-damage detection using pre-and post-seismic high-resolution satellite stereo imagery: A case study of the May 2008 Wenchuan earthquake”. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 68, pp. 13–27.
- Vaswani, Ashish et al. (2017). “Attention is all you need”. In: *arXiv preprint arXiv:1706.03762*.
- Vetrivel, Anand et al. (2015). “Identification of damage in buildings based on gaps in 3D point clouds from very high resolution oblique airborne images”. In: *ISPRS journal of photogrammetry and remote sensing* 105, pp. 61–78.

- (2016). “Identification of structurally damaged areas in airborne oblique images using a visual-bag-of-words approach”. In: *Remote sensing* 8.3, p. 231.
- Vinogradova, Kira, Alexandr Dibrov, and Gene Myers (2020). “Towards interpretable semantic segmentation via gradient-weighted class activation mapping (student abstract)”. In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 34. 10, pp. 13943–13944.
- Vu, Tuong Thuy, Masashi Matsuoka, and Fumio Yamazaki (2005). “Detection and animation of damage using very high-resolution satellite data following the 2003 Bam, Iran, earthquake”. In: *Earthquake Spectra* 21.1_suppl, pp. 319–327.
- Wang, Lei et al. (2019a). “Graph attention convolution for point cloud semantic segmentation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10296–10305.
- Wang, Peng-Shuai et al. (2017). “O-cnn: Octree-based convolutional neural networks for 3d shape analysis”. In: *ACM Transactions On Graphics (TOG)* 36.4, pp. 1–11.
- Wang, Yue et al. (2019b). “Dynamic graph cnn for learning on point clouds”. In: *Acm Transactions On Graphics (tog)* 38.5, pp. 1–12.
- Weber, Ethan and Hassan Kané (2020). “Building disaster damage assessment in satellite imagery with multi-temporal fusion”. In: *arXiv preprint arXiv:2004.05525*.
- Wei, Xin, Ruixuan Yu, and Jian Sun (2020). “View-gcn: View-based graph convolutional network for 3d shape analysis”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1850–1859.
- World Bank - ImageCat Inc. (Aug. 2010a). *POST-DISASTER BUILDING DAMAGE ASSESSMENT USING SATELLITE AND AERIAL IMAGERY INTERPRETATION, FIELD VERIFICATION AND MODELING TECHNIQUES*. Available at <https://www.geonode-gfdrrlab.org/documents/448/link> (2022/11/30).
- (July 2010b). *RIT Haiti Earthquake LiDAR dataset*. Distributed by Open-Topography. <https://doi.org/10.5069/G96Q1V50>. Accessed: 2022-12-01.
- Wu, Wenxuan, Zhongang Qi, and Li Fuxin (2019). “Pointconv: Deep convolutional networks on 3d point clouds”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9621–9630.
- Wu, Zhirong et al. (2015). “3d shapenets: A deep representation for volumetric shapes”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1912–1920.
- Xiu, Haoyi, Takayuki Shinohara, and Masashi Matsuoka (2019). “Dynamic-scale graph convolutional network for semantic segmentation of 3d point cloud”.

- In: *2019 IEEE International Symposium on Multimedia (ISM)*. IEEE, pp. 271–2717.
- Xiu, Haoyi et al. (2020). “Collapsed Building Detection Using 3D Point Clouds and Deep Learning”. In: *Remote Sensing* 12.24, p. 4057.
- Xiu, Haoyi et al. (2021a). “Enhancing Local Feature Learning for 3D Point Cloud Processing using Unary-Pairwise Attention”. In: *32nd British Machine Vision Conference 2021, BMVC 2021, Online, November 22-25, 2021*, p. 49.
- Xiu, Haoyi et al. (2021b). “Preliminary study on development of LiDAR point cloud dataset for earthquake damage mapping by 3D machine learning”. In: *17th World Conference on Earthquake Engineering*.
- Xiu, Haoyi et al. (2022a). “Enhancing Local Feature Learning for 3D Point Cloud Processing using Unary-Pairwise Attention”. In: *arXiv preprint arXiv:2203.00172*.
- Xiu, Haoyi et al. (2022b). “Statistical analysis and modeling to examine the exterior and interior building damage pertaining to the 2016 Kumamoto earthquake”. In: *Earthquake Spectra* 38.1, pp. 310–330.
- Xiu, Haoyi et al. (2023). “DS-Net: A dedicated approach for collapsed building detection from post-event airborne point clouds”. In: *International Journal of Applied Earth Observation and Geoinformation* 116, p. 103150.
- Xu, Mutian et al. (2021). “PAConv: Position Adaptive Convolution with Dynamic Kernel Assembling on Point Clouds”. In: *arXiv preprint arXiv:2103.14635*.
- Yamazaki, Fumio, T Thuy Vu, and Masashi Matsuoka (2007). “Context-based detection of post-disaster damaged buildings in urban areas from satellite images”. In: *2007 Urban Remote Sensing Joint Event*. IEEE, pp. 1–5.
- Yamazaki, Fumio, Yoshihisa Yano, and Masashi Matsuoka (2005). “Visual damage interpretation of buildings in Bam city using QuickBird images following the 2003 Bam, Iran, earthquake”. In: *Earthquake Spectra* 21.1_suppl, pp. 329–336.
- Yan, Xu et al. (2020). “Pointasnl: Robust point clouds processing using nonlocal neural networks with adaptive sampling”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5589–5598.
- Yang, Jiancheng et al. (2019). “Modeling point clouds with self-attention and gumbel subset sampling”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3323–3332.
- Yang, Ze and Liwei Wang (2019). “Learning relationships for multi-view 3D object recognition”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 7505–7514.
- Yi, Li et al. (2016). “A scalable active framework for region annotation in 3d shape collections”. In: *ACM Transactions on Graphics (ToG)* 35.6, pp. 1–12.

- Yonglin, Shen, Wu Lixin, and Wang Zhi (2010). “Identification of inclined buildings from aerial lidar data for disaster management”. In: *2010 18th International Conference on Geoinformatics*. IEEE, pp. 1–5.
- Yousefhussien, Mohammed et al. (2018). “A multi-scale fully convolutional network for semantic labeling of 3D point clouds”. In: *ISPRS journal of photogrammetry and remote sensing* 143, pp. 191–204.
- Yu, Tan, Jingjing Meng, and Junsong Yuan (2018). “Multi-view harmonized bilinear network for 3d object recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 186–194.
- Yu, Xumin et al. (2022). “Point-bert: Pre-training 3d point cloud transformers with masked point modeling”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 19313–19322.
- Zhang, Zhiyuan, Binh-Son Hua, and Sai-Kit Yeung (2019). “Shellnet: Efficient point cloud convolutional neural networks using concentric shells statistics”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1607–1616.
- Zhao, Hengshuang et al. (2019). “Pointweb: Enhancing local neighborhood features for point cloud processing”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5565–5573.
- Zhao, Hengshuang et al. (2021). “Point transformer”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 16259–16268.