SPATIALLY CONTEXT-AWARE 3D DEEP LEARNING FOR GEOSPATIAL OBJECT DETECTION

by

Tianyang Chen

A dissertation submitted to the faculty of
The University of North Carolina at Charlotte
in partial fulfillment of the requirements for
the degree of Doctor of Philosophy in
Geography

Charlotte

2024

Approved by:

Dr. Wenwu Tang

Dr. Craig Allan

Dr. Gang Chen

Dr. Shen-En Chen

ABSTRACT

TIANYANG CHEN. Spatially Context-Aware 3D Deep Learning for Geospatial Object Detection.
(Under the direction of DR. WENWU TANG)

This dissertation explores the intersection of Geographic Information Science (GIScience) and Artificial Intelligence (AI), specifically focusing on the enhancement of 3D deep learning models by spatial principles for understanding 3D geospatial data. With the rapid advancement in geospatial technologies and the proliferation of 3D data acquisition methods, there is a growing necessity to improve the capability of AI models to interpret complex 3D geospatial data effectively. This work seeks to leverage spatial principles, particularly spatial autocorrelation, to address the challenges pertaining to 3D geospatial object detection.

The research is structured around three pivotal questions: the utility of spatial autocorrelation features for understanding 3D geospatial data, the approach to derive content-adaptive spatial autocorrelation features, and the enhancement of post-processing in the task of 3D geospatial object detection by spatial interpolation. Through a series of experiments and model developments, this dissertation demonstrates that incorporating spatial autocorrelation features, such as semivariance, significantly enhances the performance of 3D deep learning models in geospatial object detection. A novel spatial autocorrelation encoder is introduced, integrating spatial contextual features into the 3D deep learning workflow and thereby improving accuracy in detecting objects within complex urban and natural environments. Further, the dissertation delves into the challenges brought by data partitioning and sampling in large-scale 3D point clouds, as evidenced in the DeepHyd project focusing on the detection of hydraulic structures (i.e., bridge and its components). The findings highlight the critical role of spatial dependency patterns in optimizing object detection accuracy and pave the way for future improvement of the 3D deep learning frameworks.

ACKNOWLEDGMENTS

I am profoundly grateful to my advisor and committee chair, Dr. Wenwu Tang, for his exceptional guidance and mentorship throughout my PhD journey. Dr. Tang has been much more than a scholar; he has truly shown the role of a mentor in every aspect, profoundly influencing my understanding not only of my field but also enriching my personal development in mind, humanity, and ethics. His support during the challenging time of the pandemic was important to me in terms of spiritual and physical, for which I am eternally thankful.

I extend my sincere thanks to my committee members, Drs. Craig Allan, Gang Chen, and Shen-En Chen, for their insightful comments and suggestions which have significantly contributed to the refinement of this dissertation. Their expertise and thoughtful guidance have been instrumental in shaping both the academic rigor and the direction of my research. Each of them has provided unique perspectives that have challenged me to think more deeply and critically, enhancing not only my work but also my growth as a scholar. Their commitment to excellence and their willingness to share their profound knowledge have been invaluable to my academic journey. Additionally, I want to thank Dr. Wei-Ning Xiang, whose unique perspective and philosophical approach to geography, encapsulated by his mantra to "sharpen your mind," have dramatically broadened my understanding of the discipline and deeply impacted my intellectual growth. He has been training me to be a thinker and a practical researcher, challenging me to exceed my boundaries and strive for new levels of scholarly inquiry.

Special thanks go to Navanit Sri Shanmugam for his assistance in collecting LiDAR data for my dissertation, and to my colleagues at the Center for Applied GIScience, Zachery Slocum and Tarini Shukla, whose friendship and insights were invaluable.

My deep appreciation extends to my mother Suying Zhang and my father Yingsheng Chen, whose endless encouragement and support have been pivotal throughout my journey. They have been the cornerstone of my life. Their unchanged belief in my abilities and their constant love have supported me against the challenges of this rigorous academic endeavor. I hold dear the memories of my late grandmothers, Aizhi Ma and Fuqin Feng, who passed away during my PhD studies. They always hoped to witness any of my achievements in my life, and their belief in my potential continues to inspire me.

Last, I must express my appreciation to my girlfriend, Siyuan Zhao. Meeting you was the luckiest thing during my PhD journey. Your presence during my hardest time has not only brought light and color to my cloudy world but has also made me appreciate you more, for the way you are and the way you make me feel. I could not have completed this dissertation without your love and support. To our future children who may one day read this: know that your mother has been my inspiration and my strength. Her love makes me a better person every day, and it is with this love that I hold dear for her and for all of you.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

2D	Two dimensional
3D	Three dimensional
AI	Artificial Intelligence
AA	Average Accuracy
ANN	Artificial neural network
CE	Cross Entropy
CNN	Convolutional neural network
FP	False Positive
FN	False Negative
GIS	Geographic information systems
GNSS	Global navigation satellite system
GPU	Graphic processing unit
GD	Gradient descent
GeoAI	Geospatial Artificial Intelligence
HPC	High Performance Computing
IoU	Intersection over Union
I/O	Input and output
IK	Indicator Kriging
ISPRS	International Society for Photogrammetry and Remote Sensing
KNN	K nearest neighbor
LiDAR	Light Detection and Ranging
MSE	Mean squared error
MSG	Multi-scale point grouping
MRG	Multi-resolution point grouping

MLP	Multi-layer perceptron
mIoU	Mean Intersection over Union
OA	Overall Accuracy
OK	Ordinary Kriging
RGB-D	Red, green, and blue with depth
RNN	Recurrent neural network
STEM	Science, technology, engineering, and mathematics
SIFT	Scale-invariant feature transform
SGD	Stochastic gradient descent
SSG	Single-scale point grouping
S3DIS	Stanford Large-Scale 3D Indoor Spaces Dataset
TP	True Positive
TN	True Negative
UK	Universal Kriging

1 INTRODUCTION

"Geographers study places" (Tuan 1977, p. 3). Geospatial technologies represented by remote sensing (e.g., LiDAR), and Global Navigation Satellite System (GNSS) (e.g., GPS, Beidou, and GLONASS) enable us to acquire geospatial big data. The data collected in space have been used to improve our understanding of our place by analyzing the data, as well as making it a better place by planning - "[p]lanners would like to evoke a sense of place" (Tuan 1977, p. 3). Geographic information systems (GIS) and Geocomputation are adopted by researchers and practitioners for geospatial analytics, featured by geovisualization (cartography), spatial statistics, spatial optimization, and spatial simulation. Conventionally, GIS application is operated based on a two-dimensional (e.g., latitude and longitude) geospatial dataset to analyze and visualize the relationship between the geospatial features (e.g., point, line, and polygon). Even though 3D data can provide more details by adding the other dimension to represent the shape of an object as to how we sense the world in reality, 3D data are more likely to be used for geovisualization not for analytics in early studies. By 2002, Zlatanova, Rahman and Pilouk (2002) reviewed GIS related software (i.e., OpenGIS) at that time where he addressed that the significant 3D progress mainly contributed to the presentation of the data. Around a decade ago, Goodchild (Goodchild 2010) reviewed the progress of GIS from the 1990s. He claimed that the truly 3D GIS (as opposed to 2.5D, such as a digital elevation model) can be very useful for applications within complex internal structures, such as mines and buildings; however, the progress is hindered because of lacking cost-effective indoor data acquisition and positioning technologies (compared with efficient 2D remote sensing data collection, and outdoor GPS positioning). There is an emerging trend of 3D GIS especially with the increase of availability of 3D data, enhancing capability in data transferring (i.e., 5G network), development in computing resource (i.e., quantum computer, cutting-edge

GPU), and evolving virtual reality technique (or augmented reality technique), where they cannot be imagined in the previous periods.

Acquisition of the 3D point cloud data for indoor and outdoor scenes becomes increasingly efficient, productive, and affordable with the rapid development of the geospatial technologies featured by remote sensing and global navigation satellite system (GNSS). A variety of types of sensors including LiDAR instruments, 3D scanners, RGB-D cameras can be mounted to tripods, mobile vehicles, and drones to facilitate the 3D data acquisition for different applications, as well as different purposes. The collected 3D data can provide productive information of real-world objects including geometry, location, and spectrum (e.g., color). 3D data is designed to be represented in different ways (e.g., point cloud, voxel, and mesh) as per the different sensors or in terms of different utilizations. Many 3D data benchmarks had become available in 2010s, containing challenges such as 3D object detection, 3D classification, and 3D semantic segmentation. Early 3D object reconstruction challenges are provided by *ISPRS Benchmark Test on Urban Object Detection and Reconstruction* (Rottensteiner et al. 2014). RGB-D (i.e., color and depth) imagery are essentially geospatial data collected by RGB-D sensors, featured by low-cost but low poor resolutions compared to laser scanners (i.e., LiDAR). The corresponding datasets include NYU (New York University) Kinect dataset (Silberman et al. 2012) and Sun RGB-D benchmark (Song, Lichtenberg and Xiao 2015). Early 3D point cloud benchmark dataset are acquired by mobile device such as the Sydney Urban Objects benchmark (De Deuge et al. 2013), the Paris-rue-Madame database (Serna et al. 2014), and TerraMobilita/iQmulus urban point cloud analysis benchmark (Vallet et al. 2015). Static LiDAR (i.e., terrestrial LiDAR) is able to collect higher resolution point cloud than the mobile one; Semantic3D benchmark (Hackel et al. 2017) is such a high-density point cloud dataset for scenes of urban and rural areas. All of the datasets above opened challenges to AI researchers and they served as a platform for competition purposes among the AI models; moreover, they also played a role of qualified sources for machines to gain knowledge so that they could be better applied to different applications (e.g., by transfer learning technique).

Artificial Intelligence (AI), imitating human intelligence to assist human solving problems in an intelligent way, has intrigued researchers from many STEM (science, technology, engineering, and mathematics) domains from academia over the years, benefiting the understanding of sophisticated phenomena, assisting complex decision making, spurring evolution of the contemporary society, and facilitating human life. Other than academia, it also plays an integral role in the modern industry - the emergence of the Fourth Industrial Revolution claimed by Schwab (2017) who is the founder and executive chairman of the World Economic Forum, along with other state-of-the-art technologies such as the Internet of Things (Philbeck and Davis , Magomadov , Atzori, Iera and Morabito 2017), and geospatial technologies (Yusoff, Ramli and Al-Kasirah 2021). As per an online survey - Global Survey: The state of AI in 2020 - by Mckinseys & Company (2020), 1,151 over 2,395 (around 56%) participants as a representation ranging from academia to industries stating that AI are adopted in at least one functions in their organizations. With AlphaGo (Silver et al. 2016), the first computer program defeated the 9-dan expert, Lee Sedol, in a Go match in March 2016, deep learning-based AI exhibited a monstrous intelligence in this ancient human intelligent game outperforming traditional machine learning algorithms designed for the Go game. Deep learning as a sub-domain of AI based on deep neural network algorithms have been representing frontier capabilities of AI, not only performing high intelligence but also waiving the need of human interaction during the training process (Bengio, Lecun and Hinton 2021, Zlochower et al. 2020). The rapid evolution of computing GPUs and availability of big data are assumed to be the main reasons triggering the success of deep learning (Ioannidou et al. 2017). Impressive progress has been made using deep learning in the recent decade in natural language processing tasks (LeCun, Bengio and Hinton 2015, Young et al. 2018), such as machine translation (Singh, Sharma and Nagesh 2017), speech recognition (Nassif et al. 2019), and image recognition (Minaee et al. 2021).

Is there a way to make machines understand 3D data as "...[n]onhuman animals also have a sense of territory and of place" (Tuan 1977, p. 4)? Deep learning-based algorithms driven by 3D data are just designed to contribute to this thread, which are featured by a series of tasks comprising: 3D object classification, 3D object detection, and 3D semantic segmentation (i.e., part segmentation, and scene parsing). 3D object classification is to make the machine recognize what the 3D data represents. Object detection in 3D context can further detect the bounding box of an object in 3D data. 3D semantic segmentation is to further label each point from a point cloud. These algorithms are designed to understand different 3D representations: voxel (i.e., 3D pixel), 2D views of 3D data, and 3D point cloud to satisfy the need of different scenarios. Even though there are other 3D representations like mesh, which is commonly used for 3D visualization but rarely seen as the input of deep learning algorithms in my literature review.

Early deep learning methods in 3D context tried to exploit the descriptors extracted from 3D data. For example, Liu, Salzmann and He (2014) use SIFT (Scale-invariant feature transform) descriptors to generate the 3D features from depth images, and use them to prepare training data fed to a Deep Belief Networks, introduced by Hinton (2009). Early such studies using deep learning in 3D context adopted voxel, 2D views of 3D data, and mesh to represent 3D data (Ioannidou et al. 2017) in order to make the data structured so that they can be fed to neural networks. 3D ShapeNets (Wu et al. 2014) and VoxNet (Maturana and Scherer 2015), represent early methods to work on 3D representations - consuming a 3D voxel grid as input to a 3D convolutional neural network, where a 3D filter was applied to extract local features of the 3D data, outperforming the other state-of-the-art methods at that time (i.e., early to mid-2010s). There is also a study trying to take mesh as input to generate 3D features like Mesh Convolutional Restricted Boltzmann Machine (Han et al. 2016). Adopting multiple 2D views of a 3D object from different directions is another way to represent the 3D object. Zhu et al. (2014) represents one of the early approaches that take multiple 2D views as input to CNN-based architectures. Multi-View CNN (Su et al. 2015) as a representation of the methods on 2D views of 3D data outperformed other tested methods in ModelNet40. The above approaches, no matter taking descriptors and multiple 2D views, or the voxel grids and mesh of 3D objects as input to the types of deep neural networks, require manipulations on the 3D data collected by the sensors (e.g., LiDAR) so that the converted data can fit the structures of input of types of deep neural networks. Such manipulations can result in either loss of local feature or introducing more error to the original data; moreover, the manipulations, especially voxel-based method, make the computational cost of processing 3D data a notable bottleneck (Ioannidou et al. 2017, Qi et al. 2017).

The emerging era of deep learning directly on 3D point clouds has been triggered since 2017 by introducing PointNet (Qi et al. 2017) featured by its capability of directly using point clouds as input. Rather than previous methods on multiple 2D views, voxel grids or mesh, which require data transformation from point cloud, PointNet firstly takes raw point cloud as input making it as an end-to-end method, and mitigating the computational challenge when scaling up in the size and resolution of the data (Qi et al. 2017).

In the past decade, deep learning has also begun to be adopted by geographers to gain meaningful information by exploring the spatial and temporal data (Goodchild and Li 2021, Li 2021, Reichstein et al. 2019). The research domain at the junction between geography and AI, termed as GeoAI nowadays (Goodchild and Li 2021), can be traced back to the late 20th century. Early pioneers, (Smith 1984), (Couclelis 1986), especially (Openshaw 1992), and (Openshaw and Openshaw 1997), introduced AI to solve geospatial problems and brought the discussions of potential impact of AI in geographic theories and practices. A 'new' quantitative revolution had been witnessed in the 1990s in geoscience, which emphasized the necessity of data driven AI approaches in solving geospatial problems (Openshaw 1992). The increasing amount of geospatial big data nowadays demands the ability of GeoAI to solve geospatial problems by taming computational challenges brought by geospatial big data. Thus, some studies treat high-performance computing as one pier of GeoAI (Li 2021, VoPham et al. 2018). Rather than from AI to geography (Openshaw and Openshaw 1997), GeoAI also calls for developing AI by incorporating geographical principles (e.g., spatial dependency, and spatial heterogeneity) (Goodchild and Li 2021). As the developments in geospatial technologies, such as remote sensing, cutting edge devices (e.g., drone, and mobile robot) and sensors (e.g. RGB-D camera, and LiDAR) have been making it more effective, accurate, and cost-efficient to acquire 3D geospatial data (e.g. geometric information, color information, intensity) of real-world objects. Under the demand of cutting-edge applications (e.g., vehicle autonomous), deep learning on 3D geospatial data (a.k.a 3D deep learning), which makes machines understand 3D geospatial data, has been becoming ever more important. Plenty of studies in 3D deep learning have been published in the domain of computer science but few have been witnessed in the research area of GeoAI yet. Understanding 3D geospatial data, especially from the perspective of spatial principles (e.g., spatial dependency, and spatial heterogeneity) can be one of the most important focuses in the state-of-the-art GeoAI.

This dissertation will contribute to the development of GeoAI with a focus on making deep learning models better understand 3D geospatial data by leveraging spatial principles. The primary research questions are as follows: 1) whether spatial autocorrelation features are helpful for 3D deep learning models to understand 3D geospatial data? 2) How to extract effective spatial autocorrelation features by using deep neural networks? 3) How to leverage spatial principles to address the challenge pertaining to the framework of 3D geospatial object detection. The corresponding study objectives are listed below:

(1) Examine the effectiveness of semivariance as a representation of spatial autocorrelation-based features in 3D geospatial object detection.

(2) Develop a deep learning-based framework leveraging spatial dependency principle to extract spatial autocorrelation features based on pairwise differences within a local neighborhood.

(3) Improve current framework of 3D deep learning-based geospatial object detection from a metamodel perspective with a focus on using spatial interpolation in the post-processing.

In the following chapters, I will briefly review the background of deep learning including basic concepts, basic components, and related problems in Chapter 2. In Chapter 3, I evaluated the effectiveness of spatial autocorrelation features in understanding 3D geospatial data by control experiments from a data perspective (a.k.a taking semivariance as additional input). In Chapter 4, I developed a deep learning-based approach to derive spatial autocorrelation features for enhancing the performance of the 3D geospatial object detection. In Chapter 5, I improved the current 3D deep learning-based framework for 3D geospatial object detection by integrating spatial interpolation in the post-processing, where hydraulic structure detection is used as an exemplary study case. Chapter 6 is for overall conclusions and potential future work.

2 LITERATURE REVIEW

2.1 Background of Neural Networks

2.1.1 Conception of neural networks

Artificial neural network (ANN) is a method in the machine learning domain, which is originally inspired by the communication (see Figure 2.1 as a demonstration) between neurons within a nervous system from the human brain (McCulloch and Pitts 1943).

> "The nervous system is a net of neurons, each having a soma and an axon. Their adjunctions, or synapses, are always between the axon of one neuron and the soma of another. At any instant a neuron has some threshold, which excitation must exceed to initiate an impulse." (McCulloch and Pitts 1943, p. 101)

An artificial neural network also consists of a bunch of neurons with a soma (i.e., sum function), an axon (i.e., activation function), and synapses (i.e., weighted connections). Functioning similar to a neuron gets excitation and sends out an electrical impulse to other neurons through synapse in the nervous system, a neuron in ANN will transmit the output to other neurons via connections if it is activated or fired by input values in particular threshold defined by the activation function.

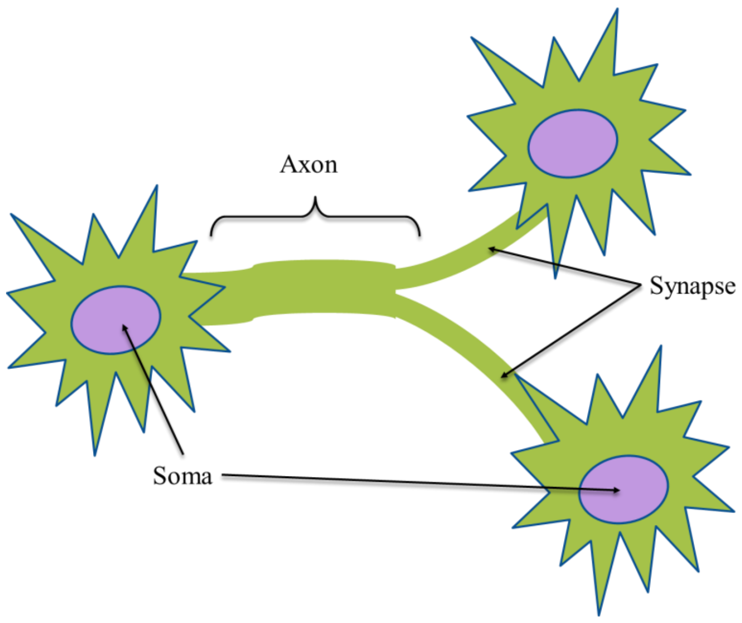


Figure 2.1. Illustration of a neuron consisting of a soma, an axon, and several synapses.

2.1.2 History of neural networks

Schmidhuber (2015) conducted a systematic review on development of deep learning with a focus on artificial neural networks. A timeline is drawn based on this systematic review as shown in Figure 2.2.

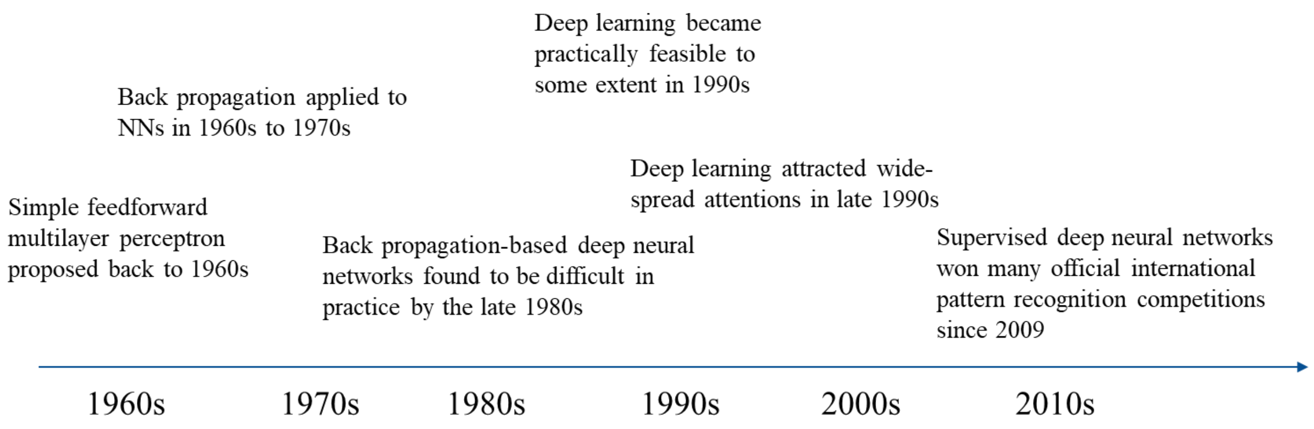

Deep learning became practically feasible to some extent in 1990s
Back propagation applied to NNs in 1960s to 1970s
Deep learning attracted wide-spread attentions in late 1990s
Simple feedforward multilayer perceptron proposed back to 1960s
Back propagation-based deep neural networks found to be difficult in practice by the late 1980s
Supervised deep neural networks won many official international pattern recognition competitions since 2009
1960s
1970s
1980s
1990s
2000s
2010s

Figure 2.2. Timeline of development of artificial/deep neural networks.

In 2001, Doug Laney from META Group (latter renamed to Gartner) firstly defined big data from a three-dimensional perspective: “Big data is high volume , high velocity , and/or high variety information assets that require new forms of processing to enable enhanced decision-making, insight discovery and process optimization” (Laney 2001, p. 67).

In 2009, Dr. Fei-Fei Li, an AI professor from Stanford University, launched ImageNet, which is an image classification benchmark with over 14 million labeled images. “Our vision was that big data would change the way machine learning works” said by Dr. Li (Li, Deng and Li 2009).

The speed of GPUs gained a dramatic increase by 2011, making it possible to train convolutional neural networks directly, which was never ever computationally feasible. With the increasing computing speed, deep learning has been outperforming other machine learning algorithms by its essential advantages regarding efficiency and accuracy. AlexNet, developed by Krizhevsky, Sutskever and Hinton (2012), is a convolutional neural network. It achieved notable success in image recognition competitions during 2011 and 2012. The network employed rectified linear units (ReLUs) as activation functions. This approach helped speed up training and tackle the vanishing/exploding gradient problem.

2.2 Basic Components of Neural Networks.

In this section, I am going to introduce the basic conceptual components of a neural network as well as embedded mathematics. I will focus on a typical neural network used for classification problems, which predicts a class label for a given input.

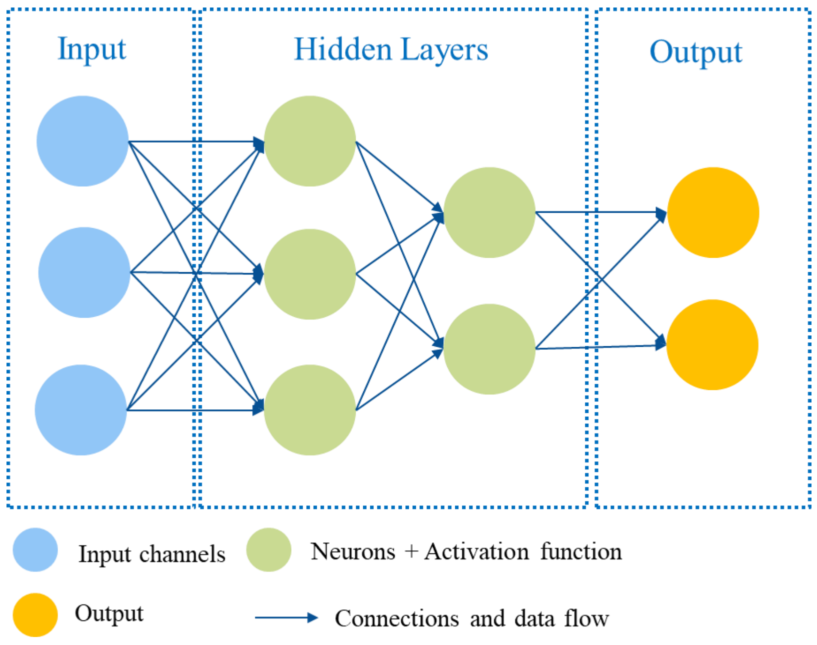


Figure 2.3. Illustration of the architecture of a feed-forward neural network (a.k.a multilayer perceptron) with two hidden layers.

As shown in Figure 2.3, a neural network is composed of an input layer, several hidden layers, and an output layer. In the input layer, each sphere represents a channel as well as a feature channel of the input data. For example, the basic three channels of a spatial point can be x, y and z coordinates in a space with respect to a coordinate system. A hidden layer comprises a series of neurons as well as corresponding activation functions following the neurons. The outputs are the probability of classes, into which each input is predicted to fall.

Basically, a neural network can be denoted as a function shown as follows:

$$C = f(X) \tag{2.1}$$

$$X = [x_1, x_2, x_3 \dots x_m] \tag{2.2}$$

$$C = [c_1, c_2, c_3 \dots c_n] \tag{2.3}$$

where X is the input with m feature channels. C is the output with n classes, which contains probability values of the input that are predicted to fall in each class. f represents a neural network with a designed number of hidden layers as well as a designed number of neurons within corresponding layers.

In the rest of this section, I am going to demonstrate what operations happen in each hidden layer after the inputs are fed to the neural network.

2.2.1 Neurons

Each hidden layer of a neural network comprises a series of neurons and corresponding activation functions following each neuron. Each neuron (see Figure 2.4) is a weighted sum function to aggregate data fed into it from either an input layer or from a previous hidden layer. The operations within each neuron as well as an activation function can be denoted in the following formula:

$$Output = g(X \cdot W + b) \tag{2.4}$$

$$W = [w_1, w_2, w_3 \ldots w_m] \tag{2.5}$$

where g is an activation function associated with this neuron, W is a set of weights separately for each input X associated with this neuron. Connections between the input to a neuron and the neuron has a set of weights, where the number of weights is equal to the number of connections as well as the number of inputs. b is a bias value to be added to the weighted sum value from a neuron before feeding them into an activation function. Both weight and bias can be updated during training. Therefore, the number of learnable parameters of the neuron is *(m+1)* (i.e., m weights and one bias).

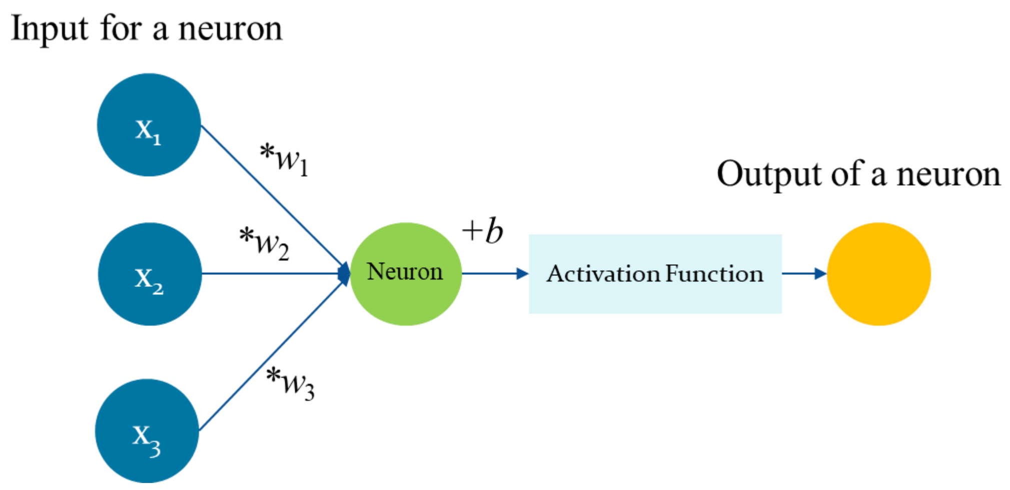


Figure 2.4. Demonstration of operations within a neuron following an activation function. The input of a neuron can be input of the neural network or output of a previous hidden layer.

2.2.2 Bias

Bias is a learnable constant per node that adds to the dot product of weights and input before fed into the activation function. Bias can shift the activation function, resulting in activating a neuron or deactivating the neuron (whether to fire a neuron). The role of bias in activation function is similar to the role of a constant value in a linear regression model, giving more flexibility to better fit a model to the observations. Bias is widely used in current neural networks, even though it is not mentioned commonly, since it becomes a default option when creating a layer for a neural network in the deep learning libraries (e.g., Pytorch).

2.2.3 Activation function

From a conceptual perspective, activation function is to decide whether a neuron is going to be activated or deactivated as per the input. In other words, it is to decide whether the input information is useful or not-that-useful to this neuron. If the information is judged as useful, it will be passed to the other neurons in the next layer. If it is not that useful, the information will be segregated, meaning it will pass 0 to the next layer, where it will not impact deeper neurons. There may not be an absolute activation and inactivation of neurons depending on the nature (i.e., the threshold of output) of an activation function. The closer the output value is, the more inactivated a neuron will be. If a neuron is inactivated or close to being inactivated. The information will be segregated. Segregation is important to a neural network that helps the model to pursue useful information instead of stuck on a not that useful information.

From a mathematical perspective, activation function can be understood as an affine transformation, which transforms the input value to a specific range of values. There are a bunch of activation functions with different target ranges of values developed for different applications.

Sigmoid/logistic

Sigmoid or logistic activation function is an S-shape function (see Figure 2.5) commonly used in prediction tasks due to the nature of its shape that the threshold of sigmoid activation function is [0,1]. The function transforms the input values to the range [0,1], which can be interpreted as probability.

$$Sigmoid(x) = \frac{1}{1+e^{-x}} \tag{2.6}$$

where x is the input.

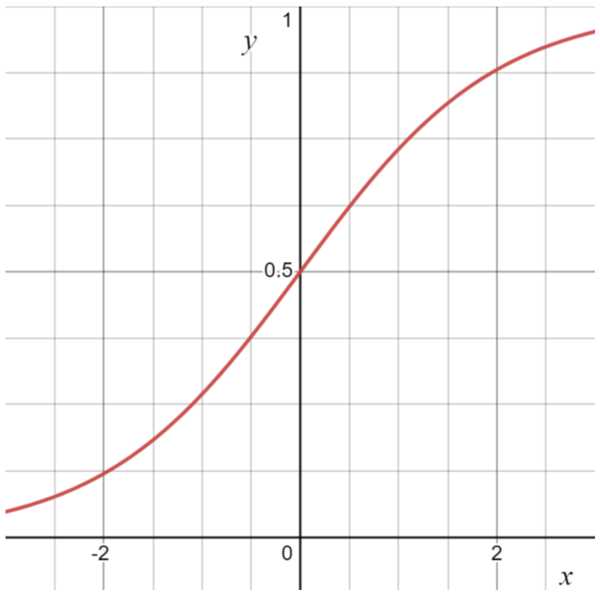


Figure 2.5. Demonstration of Sigmoid/logistic activation function.

Hyperbolic tangent function

Hyperbolic tangent function (Tanh) is the other type of S-shape function (see Figure 2.6) that transforms the input to a value ranging in [-1, +1]. This activation function is designed to be sensitive to input values within the range of -3 to +3. In other words, it does not care how far it is smaller than -3 or larger than +3. "… the hyperbolic tangent activation function typically performs better than the logistic sigmoid" (Goodfellow, Bengio and Courville 2016, p. 195).

$$Tanh(x) = \frac{e^{2x}-1}{e^{2x}+1} \tag{2.7}$$

where *x* is the input.

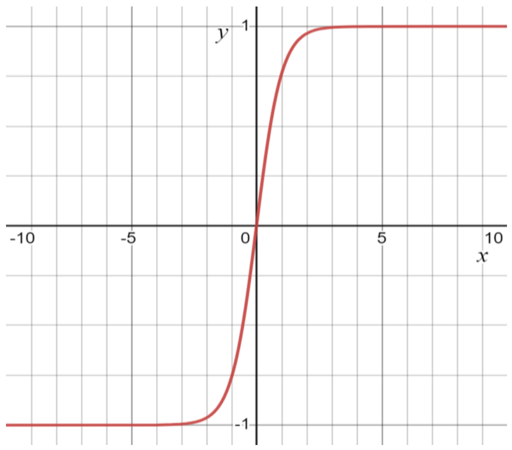


Figure 2.6. Demonstration of Hyperbolic tangent activation function.

ReLU

Rectified linear activation function (ReLU) is commonly used in image classification tasks, which outperforms other activation functions in practice of image classification tasks. ReLU (see Figure 2.7) is designed to be sensitive to positive values and ignore the negative values. In the other word, only positive values can fire a neuron with ReLU as the activation function. “In modern neural networks, the default recommendation is to use the rectified linear unit or ReLU” (Goodfellow et al. 2016, p. 174) because it can handle the problem of vanishing gradient to make neural networks deeper.

$$ReLU(x) = max(0, x) \tag{2.8}$$

where *x* is the input.

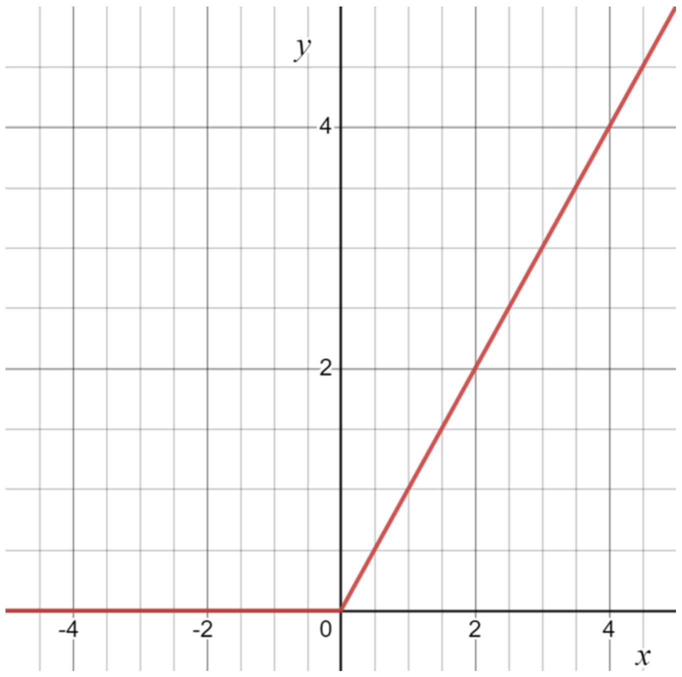


Figure 2.7. Demonstration of ReLU activation function.

Softmax

$$softmax(c_i) = \frac{\exp(ci)}{\sum_{i=1}^{n} \exp(ci)}, \; i \in [1, n] \qquad (2.9)$$

where c_i is the output of class *i*. *n* is the number of classes in a classification problem.

Softmax is a mathematical function that map a vector of values to a vector of probabilities with a sum of 1, which is commonly used as the activation function of the output layer of a neural network in order to gain a vector of probabilities of the input falling into different classes in a classification task. Rather than a max function, softmax can give a probability of the predicted label for a given input.

2.3 Basic Mechanism of Neural Network

In this section, I will introduce the mechanism of how a neural network learns based on the input by demonstrating conceptual ideas as well as mathematical foundations. There are five steps:

1. Initialize the parameters (i.e., weights and bias) of neural network.
2. Forward propagation to pass the input through the neural network.
3. Calculate the loss function.
4. Backpropagation to compute the gradients with respect to each parameter.
5. Update each parameter regarding the optimization algorithm (i.e., optimizer).

Repeat steps 3 to 5 until it meets the stop criteria.

2.3.1 Parameters Initialization

Parameter initialization gives initial values for parameters (i.e., weights and biases) of a neural network. A well-designed algorithm for initialization is important to have the neural network well trained with respect to activation function. There are a bunch of algorithms to initialize the parameters.

Zero initialization

Zero initialization, as the name implies, is to initialize all parameters as 0 before training. When initializing the parameters with 0, the backpropagation will result in the same gradients with respect to each parameter. Therefore, the parameters will be updated in the same way through the training process. In this case, the neural network will work no better than a linear model.

Random initialization

To address the problem of zero initialization, random initialization can prevent all neurons from being the same and updated in the same way. However, a random value can easily result in either exploding or vanishing gradients in backpropagation because a random initial value can be too big or too small (close to 0). Xavier initialization and Kaiming initialization (He et al. 2015), two of the most popular stochastic initialization approaches, help to reduce the vanishing and exploding gradient problem.

2.3.2 Forward propagation

Forward propagation is to describe the way to pass input through a feed-forward neural network layer by layer until the output of the neural network (as demonstrated in Figure 2.3). Each hidden layer takes the output of the previous hidden layer (the first hidden layer will take directly from the input layer). Feed-forward neural networks are different from recurrent neural networks (RNN), where connections between nodes can form a cycle. RNN, derived from the feedforward network, is used to take a sequence of inputs, which is commonly used for speech recognition or time series-related predictions. In this study, I will only focus on feedforward network since most architectures used for classification task is a feedforward network (e.g., convolution neural network)

2.3.3 Loss function calculation

In an optimization problem, an objective function is the one we want to maximize or minimize. In a neural network, we want to minimize an error function between the predicted value and ground truth. The error value, here, is also called the loss in the context of a neural network, which can be calculated based on a loss function. Loss will be calculated for each input and then they will be aggregated by sum or mean depending on the optimizer. At the end of each step (i.e., each mini-batch or the entire dataset was propagated forward until the output layer), all the errors of the input (i.e., a mini-batch or the entire dataset) will be passed to the loss function to calculate loss.

From a mathematical point of view, there are various loss functions designed for different tasks. In the rest of this section, I am going to demonstrate two classic loss functions designed separately for regression tasks and classification tasks.

Mean squared error loss function

Mean squared error (MSE) loss function is to calculate the average squared loss for each input over the entire dataset. Apparently, it is to calculate differences between two vectors. MSE is commonly used for a regression task to predict a continuous variable.

$$MSE = \frac{\Sigma(q_i - p_i)^2}{m}, i \in [1, m] \tag{2.10}$$

where q_i is the ground truth of the *i*th input and p_i is the predicted value regarding this input; *m* is the number of inputs.

Cross entropy (Logarithmic) loss function

Cross Entropy is usually applied to classification problems (i.e., binary classification and multi-class classification). Cross entropy is to calculate loss between two distributions (ground truth, and predictions) of probabilities. It should be converted to probabilities using softmax activation function before fed into cross entropy loss function.

$$CE(p, q) = -\sum_i^n [p_i * log(q_i)], p, q \in [0,1], i \in [1, n] \tag{2.11}$$

where *o* is the number of input (of the entire dataset or a mini-batch) and *n* is the number of channels; *q* is the probability of the output for each input and *p* is the ground truth. When *p* is close to q, the loss is approaching 0.

2.3.4 Gradient calculation based on backpropagation.

An optimizer or optimization algorithm in a neural network depicts an approach to update parameters within a neural network with respect to the gradient. Gradient is a set of derivatives of loss with respect to each of the parameters (i.e., weights and biases) in a neural network. An optimizer seeks to minimize the loss function by minimizing the gradient as well as the derivatives of loss with respect to the parameters.

Gradient descent (GD) is an optimization method that helps to minimize the loss function by using the product of gradient and learning rate. The prerequisite of using GD is that the function to be minimized must be differentiable and convex. Therefore, loss functions in neural networks are designed to meet the two requirements. Regular gradient descent is sensitive to the value of learning rate. The general strategy is to start with a relatively large learning rate and gradually decrease it in each step. This process that changes the learning rate from relatively large to small is scheduled. Learning rate schedulers can help to arrange this schedule in a different manner. Gradient descent can be used for many optimization problems in statistics and machine learning.

Different from least squares which optimizes the parameter and seeks the derivative equals to 0, GD can gradually find the minimum derivative by steps from the initial parameter. Therefore, GD is useful when it is not possible to solve for where the derivative is 0. The closer the parameters get to optimal values, the closer the derivative gets to 0. GD is efficient because it will take a relatively larger step when the derivative is far from 0 but it will take a smaller step when approaching 0. To avoid taking too big steps when the derivative is far from 0, a small value (i.e., learning rate) is applied to restrict the step.

The GD will stop when the step size is smaller than at a set threshold or the number of steps reach the set maximum number of steps.

$$G = \frac{1}{K}\sum(dLoss/dW_k), k \in [1, K] \tag{2.12}$$

$$W_{t+1} = W_t - G * lr, lr \in [0,1] \tag{2.13}$$

where *G* is the sum of gradients based on o input samples. *K* is the number of weights of the neural network. *dLoss/dW*$_k$ is the derivative of the loss with respect to the *k*th weight. *t* is the current iteration in a training process, *W* is a set of weights for a neural network. W_t is the current weights. W_{t+1} is the updated weight. *lr* is the learning rate.

A general step of GD is as follows:

1. Calculate the derivative of the loss function with respect to each parameter.
2. Calculate the gradient of the loss function.
3. Calculate the step sizes for each parameter.
4. Calculate the new parameters.

Loop from step 1 to step 4 until the step size is small or equal to the set value or reach the maximum number of iterations (steps).

Gradient descent calculates the gradient with respect to the loss of the entire datasets, which can result in computational issues when the entire dataset is too big. Therefore, Stochastic gradient descent (SGD) (Bottou 2010) is proposed to solve this problem. SGD will randomly select a small subset of the entire dataset (also called mini-batch) for each step that updates the parameters (i.e., weight and bias). SGD will also help to optimize the parameters when there is new data coming instead of training from scratch (if we use GD, we have to include the entire dataset), where it can use the trained parameters as initial values but optimize them based on new data only. SGD performs better than GD especially because there are a lot of parameters and big dataset because it may not be computationally feasible if using GD.

2.4 Basic Problem Related to Neural Network.

If the gradients in early layers are vanishingly small (close to zero), it makes the parameters in these layers barely update in each training step. In other words, the model is not efficiently learning within the training process. In this case, it is called a vanishing gradient problem (Hochreiter 1998).

Generally, this is due to the nature of backpropagation. The more it is early in the network, the quicker it is going to vanish. When SGD computes the gradient for a specific weight, it applies this gradient to modify the weight accordingly. Thus, the adjustment made to the weight is directly proportional to the magnitude of the gradient. If the gradient is small, the resulting update will also be extremely small. Under such circumstances, if the weight is adjusted by a trivial amount, it results in barely modifications to the weights of the neural networks. Consequently, such minor adjustments fail to propagate effectively throughout the network, rendering them ineffective in significantly reducing the loss, as the weight remains almost unchanged from its value prior to the update.

Therefore, it almost becomes a problem that the weights never actually be updated efficiently to reduce the loss or reach its optimal value, which will impact the rest of the layers in this network and prevent the ability of the network from learning.

On the other hand, it will make the parameters update too much in each training step if the gradients in early layers are extremely large. Moreover, due to the backpropagation, the more the layers are early in the network, the more the gradients are going to explode. In this case, it is called an exploding gradient problem.

2.5 Artificial Neural Networks and Deep Neural Networks

The existence of vanishing/exploding gradient problems within the traditional artificial neural networks prevents training with more layers within a neural network, which limits the ability to solve more complex problems.

There has been a revolutionary development of neural networks during 2010s, especially the related techniques and activation functions associated with neural networks, aiming to remove the barrier that prevents the neural network from going deeper. Various strategies have been used to mitigate the vanishing gradient issue, such as employing specific methods for weight initialization (Mishkin and Matas 2015), utilizing second-order optimization techniques (Martens 2010), and implementing updates on a per-layer basis (Vincent et al. 2008).

With the development in both hardware and techniques, deep learning can concatenate a series of hidden layers in a designed way for different tasks (Potok et al., 2018). Same as machine learning, deep learning can be either supervised or unsupervised. In this study, I will only focus on supervised architectures in deep learning. There are two basic supervised deep learning architectures, recurrent neural networks, and convolutional neural networks. Recurrent neural networks are commonly used in processing natural language or time series data. CNN is often used for video analysis, natural language processing, and image recognition. The image-related tasks are object detection, image classification, semantic segmentation, instant segmentation.

2.6 Convolutional Neural Network

In the following section, I am going to introduce the basic components of a Convolutional Neural Network (CNN).

The basic components of a CNN consist of an input layer, convolutional layers, pooling layers (e.g., max pooling), fully connected layers, and an output layer as shown in Figure 2.8. Using imagery as input for an example, the input layer has three dimensions, heigh, width, and feature channels (e.g., red, blue, and green). The height and width are the size of an image, and the feature channels can be either colors in RGB color space, or one gray channel. The output layers are the labels to be predicted on the input. For example, if the model is designed to predict cat or dog based on the input imagery, the output layer will be the probabilities of the input to be predicted as cat and dog, when a softmax is used as the activation function in the last hidden layer. Hidden layers can have a series of combinations of convolutional layers and pooling layers, as well as fully connected layers. Convolutional layers and pooling layers are to extract feature maps using a series of receptive fields (i.e., kennels). On the other hand, fully connected layers are designed for classification based on a high dimensional feature derived by flattening the feature maps from feature extraction layers.

In the following sections, I am going to explain the conceptual and mathematical details of each layer.

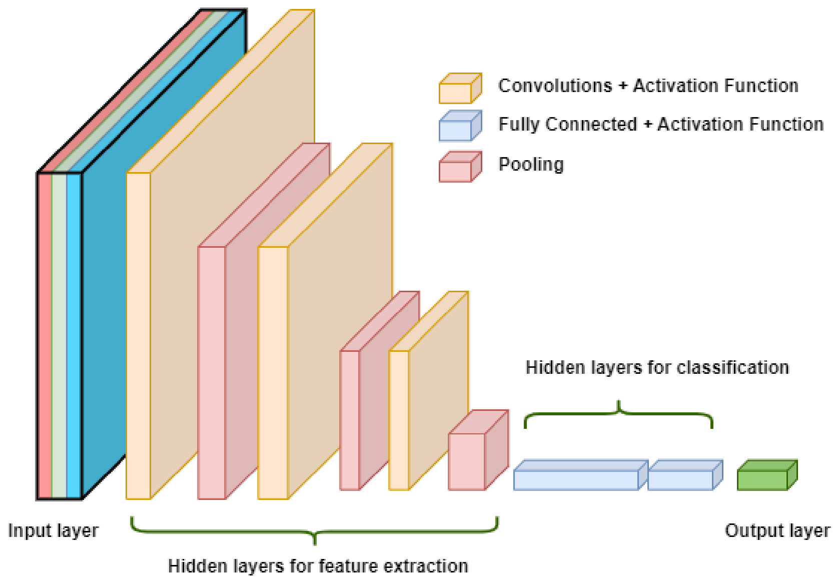

Convolutions + Activation Function
Fully Connected + Activation Function
Pooling
Hidden layers for classification
Input layer
Output layer
Hidden layers for feature extraction

Figure 2.8. Illustration of the architecture of a convolutional neural network. An input image with three channels is fed into the neural network. The output layer contains the probabilities of each class the input can fall into.

2.6.1 Convolutional layer

A convolution layer consists of a series of kernels and an activation function associated with each kernel. A convolutional operation (see Figure 2.9) is to aggregate the information of the data fed into the convolutional layer with respect to a designed stride and padding. Basically, a convolutional operation can be denoted as the following formula:

$$Output = g\ (Sum\ (W.*X) + b) \tag{2.14}$$

$$W = [w_1, w_2, w_3,\ \ldots\ w_{n'}] \tag{2.15}$$

$$X = [x_{11}, x_{12}, x_{13},\ \ldots\ x_{mn'}] \tag{2.16}$$

where g is an activation function associated with a kernel. W represents the weights of the kernel; X are the values of input that the current kernel covers. m is the number of channels of the input. n' is the number of cells in a kernel. b is the bias associated with each kernel.

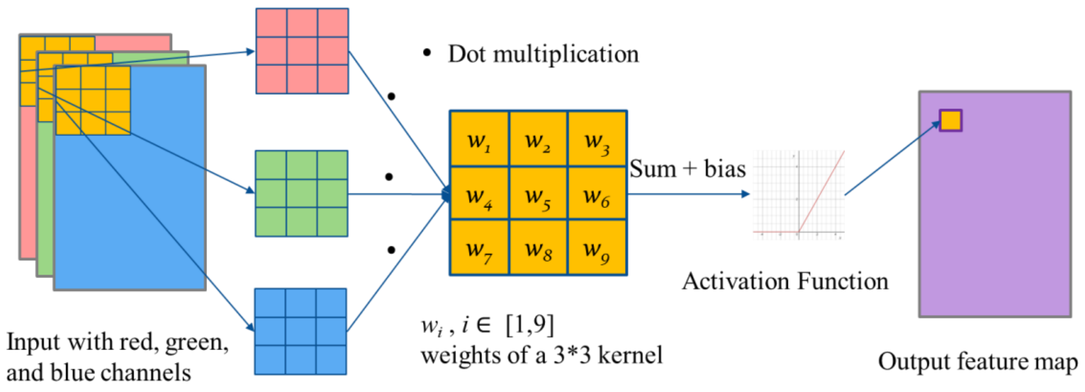

Dot multiplication
w_1 w_2 w_3 w_4 w_5 w_6 w_7 w_8 w_9
Sum + bias
Activation Function
Input with red, green, and blue channels
w_i , $i \in$ [1,9] weights of a 3*3 kernel
Output feature map

Figure 2.9. Demonstration of convolutional operation with a 3*3 kernel along with an activation function on a particular cell. A feature map can be derived by conducting the convolutional operation on each cell.

A convolutional layer conducts convolutional operations with each designed kernel on each cell along with an activation function to generate an output feature map (a.k.a., activation map) as shown in Figure 2.9. From a conceptual perspective, feature maps are representations of spatial presence of patterns or concepts (e.g., edges).

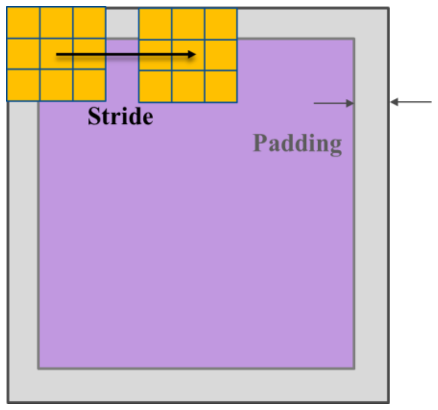


Figure 2.10. Demonstration of stride and padding in the convolutional operation.

Rather than the size of a kernel, there are the other two hyper parameters to configure when conducting the convolutional operation, stride and padding (as shown in Figure 2.10). Stride and padding are adjustable parameters that can maintain the dimensions of the output feature map equal to those of the input. The stride refers to the distance by which a kernel shifts across the input with each step, while padding involves appending extra rows and columns of zeroes to the input, ensuring the size of the output remains consistent with that of the input.

2.6.2 Pooling

Pooling is one of the layers of a CNN following a convolutional layer, which is an optional layer in a CNN-based architecture design of a neural network. Features used for a classifier (i.e., fully connected layer in the context of CNN) represented by an output feature map from a convolutional layer can be sensitive to the location of the features. To address the sensitivity, a typical approach is down sampling the feature maps. Conceptually, a down-sampled feature map makes it robust to changes in terms of the locations of a feature from a feature map. Theoretically, a pooling layer can make feature maps local translation invariance (Goodfellow et al. 2016).

As explained by (Goodfellow et al. 2016, p. 342), "…pooling helps to make the representation become approximately invariant to small translations of the input. Invariance to translation means that if we translate the input by a small amount, the values of most of the pooled outputs do not change". Pooling layers come in two primary forms, max pooling and average pooling. Max pooling selects the maximum value from each patch of the input feature map, while average pooling computes the average value of the elements in each patch, thereby reducing the spatial dimensions of the input feature map. However, "it's more informative to look at the maximal presence of different features than at their average presence" (Ketkar and Santana 2017, p. 129).

A max pooling operation can be simply denoted by the following equation. I demonstrated the pooling operation in Figure 2.11.

$$output = Max(X) \tag{2.17}$$

$$X = [x_1, x_2, x_3, \ldots x_n] \tag{2.18}$$

where X is the input and n is the number of input.

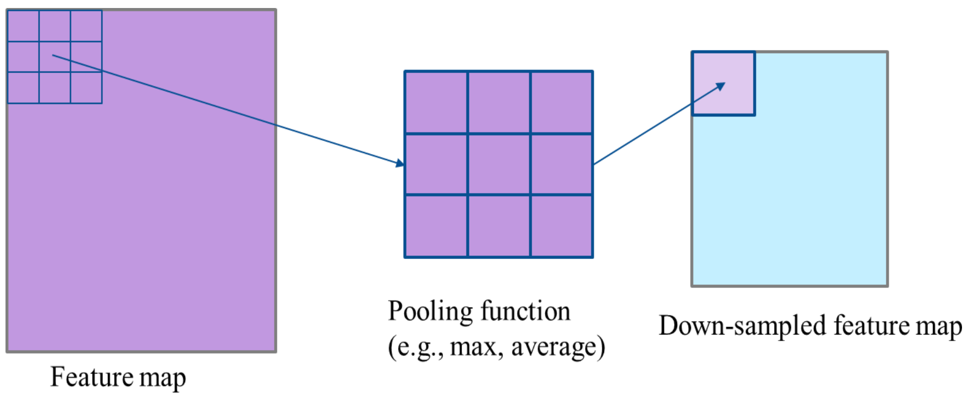


Figure 2.11. Demonstration of pooling operation in a convolutional neural network. A 3*3 pooling kernel is used for demonstration purposes.

2.6.3 Fully connected layer

Fully connected layers play a role of classifier in a CNN as shown in Figure 2.12, which take the flattened feature maps extracted by the feature extraction hidden layers (i.e., convolutional layers and pooling layers) as input and output probabilities of labels that the input is predicted to be. It is called fully connected because each feature is connected to each neuron within this layer, where it is only partially connected in the convolutional and pooling layers.

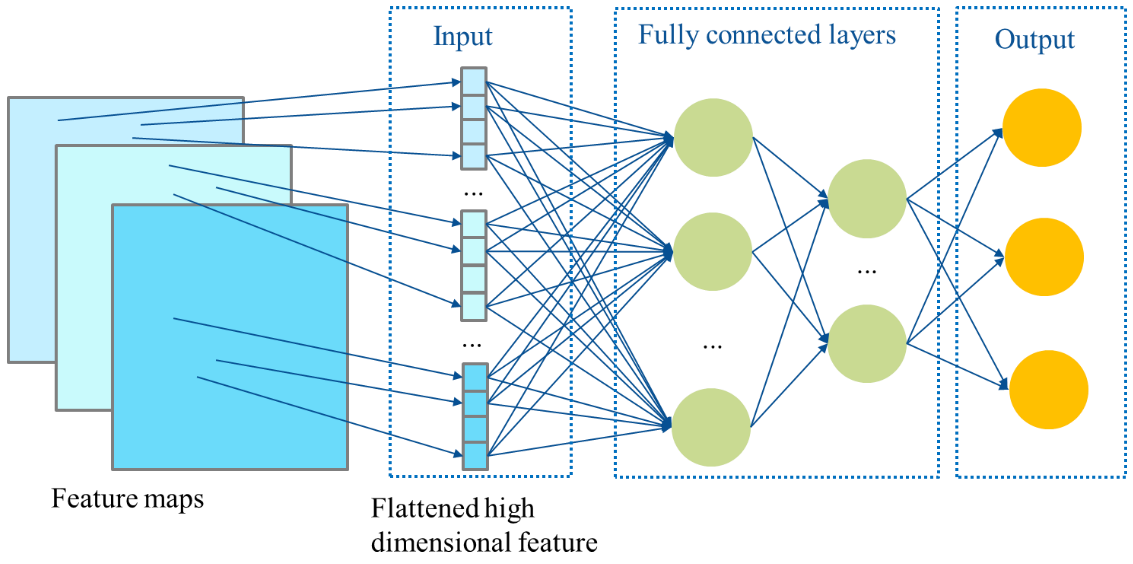


Figure 2.12. Demonstration of fully connected layer and output layer. Feature maps are from feature extraction layers (e.g., convolutional, and pooling layers).

The feature maps are flattened to be a high dimensional feature vector before fed into a fully connected layer. A fully connected layer consists of a series of neurons and an activation function associated with each neuron. The flattened features as to the fully connected layer in a CNN is the same as the input as to the hidden layers of a multilayer perceptron shown in Figure 2.3.

Reference

Bottou, L. 2010. Large-Scale Machine Learning with Stochastic Gradient Descent. In *Proceedings of COMPSTAT'2010*, 177-186. Physica-Verlag HD.

Goodfellow, I., Y. Bengio & A. Courville. 2016. *Deep Learning*. MIT Press.

Hochreiter, S. (1998) The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions. *Int. J. Uncertainty Fuzziness Knowledge Based Syst.*, 06**,** 107-116.

Ketkar, N. & E. Santana. 2017. *Deep learning with Python*. Springer.

Krizhevsky, A., I. Sutskever & G. E. Hinton (2012) Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25.

Laney, D. (2001) 3D data management: Controlling data volume, velocity and variety. *META group research note*.

Li, F.-F., J. Deng & K. Li (2009) ImageNet: Constructing a large-scale image database. *J. Vis.*, 9**,** 1037-1037.

Martens, J. (2010) Deep learning via hessian-free optimization.

McCulloch, W. S. & W. Pitts (1943) A logical calculus of the ideas immanent in nervous activity. *Bull. Math. Biophys.*, 5**,** 115-133.

Mishkin, D. & J. Matas (2015) All you need is a good init. *arXiv [cs.LG]*.

Schmidhuber, J. (2015) Deep learning in neural networks: an overview. *Neural Netw.*, 61**,** 85-117.

Vincent, P., H. Larochelle, Y. Bengio & P.-A. Manzagol. 2008. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, 1096-1103. Association for Computing Machinery.

3 EXPLICIT INCORPORATION OF SPATIAL AUTOCORRELATION IN 3D DEEP LEARNING FOR GEOSPATIAL OBJECT DETECTION

3.1 Introduction

The accelerating convergence between geographic information science (GIScience) and artificial intelligence (AI), collectively termed GeoAI (Goodchild 2022), has been changing the ways we use and interpret 3D geospatial data. In an era marked by quasi-exponential growth in such data (Li, Hodgson and Li 2018), from RGB-D images to 3D point clouds, the demand for identification and localization of 3D objects (i.e., 3D object detection) has been increasing in applications ranging from autonomous vehicles to crime scene investigations. The importance of accurate 3D object detection cannot be underestimated in these contexts. For example, accuracy in identifying and localizing 3D objects in autonomous vehicles is not only a technological achievement for researchers and practitioners but also a critical safety requirement demanded by the general public. Any small error in detecting obstacles can potentially result in accidents, posing risks to human lives. In the realm of crime scene investigations, accurate 3D object detection is essential for the faithful reconstruction of criminal events such as detecting the position of a victim, exhibits evidence from the surrounding environment, which can help corresponding agencies accurately interpret and document criminal scenes. Deep learning has become a powerful solution in these contexts due to its outstanding performance surpassing traditional machine learning methods. Moreover, its evolution also outpacing them, given unprecedented focus and contributions by different domains (e.g., engineering, medicine, computer science, and geography).

GIScience has been able to better derive knowledge from geospatial data by using deep learning (Goodchild and Li 2021). For example, Man and Liu (2021) used deep neural network to reliably downscale air quality related variables. Duan et al (2020) adopted reinforcement learning to automate the alignment between vector data and historical maps. The origins of GeoAI can be traced back to the late 20th century when pioneers like Smith (1984), Couclelis (1986), and especially Openshaw (1992), Openshaw and Openshaw (1997) began applying AI techniques to detect patterns embedded in geographic data, contributing to the early development of spatial data science. Over time, GeoAI has evolved to not only adopt AI techniques for geographic applications but also contribute to the development of AI itself by incorporating geographic principles such as spatial dependency and spatial heterogeneity, whether it is explicitly stated or not (Goodchild and Li 2021). For example, Goodchild and Li (2021) attributed the exceptional performance of convolutional neural networks (CNN) on object detection tasks to its spatial dependency incorporated methodological design.

While deep learning has shown unprecedented capabilities in geospatial object detection from remotely sensed data, it is still an open question with respect to the handling of spatial dependencies of observed attributes (e.g., color information and LiDAR[1] intensity) within 3D geospatial data. Traditional models especially rely on the spatial properties (i.e., XYZ coordinates) of 3D data as well as the shape features embedded in them. Some of them include hand-crafted features (e.g., pre-estimated surface normal) to enhance model performance on object detection (Qi et al. 2017a, Zhao et al. 2021). Nevertheless, these studies often overlook the potential benefits of color and its spatial autocorrelation features in improving model performance. This might be due to the lack of color information for many benchmark datasets (e.g., ShapeNet[2], SemanticKITTI[3]) where a neural network architecture might not be designed to require color as input. However, color information has been increasingly incorporated in modern data acquisition

[1] LiDAR stands for light detection and ranging technology.
[2] ShapeNet: https://shapenet.org/
[3] SemanticKITTI: http://www.semantic-kitti.org/

using instruments such as LiDAR and RGB-D[4] cameras (e.g., S3DIS[5], Semantic3D[6]). Therefore, it is important to make use of, and take advantage of such observed non-spatial attributes as well as the spatial autocorrelation features inherent to this data to achieve better performance.

This study pioneered efforts to bridge this gap by explicitly incorporating spatial autocorrelation of color information into 3D deep learning models for geospatial object detection. Inspired by the success of spatial autocorrelation features in object detection on 2D geospatial data (Bian and Xie 2004, Bian and Lee 2005). We explore their potential for boosting the performance of 3D deep learning models. Our study aims to both advance the field of GeoAI by promoting the linkages between deep learning and GIScience and contribute to the ongoing development of deep learning techniques adapted for geospatial applications.

Our main contributions are:

1. Explicit incorporation of spatial autocorrelation into 3D deep learning: Our study breaks new ground by being the first to explicitly integrate spatial autocorrelation features, particularly semivariances, to enhance the performance of 3D deep learning models in geospatial object detection.
2. Novel approach of spatial autocorrelation estimation in 3D data: We propose a novel framework to estimate spatial autocorrelation features for object detection in the context of 3D deep learning, and tackle challenges posed by the nature of unstructured and unevenly distributed 3D data.
3. Insights of spatial autocorrelation as contextual information in 3D deep learning: We provide insights into the effectiveness of spatial autocorrelation features serving as context information affected by how neighborhood or extent is defined in different environments, such as indoor and outdoor settings. This contributes to the understanding of the uncertain

[4] RGB-D stands for Red, Green, Blue and Depth.
[5] S3DIS: http://buildingparser.stanford.edu/
[6] Semantic3D: http://www.semantic3d.net/

geographic context problem (Kwan 2012) with respect to 3D geospatial object detection, and the development of more robust and accurate models in the field of 3D deep learning.

The remainder of this paper is organized as follows: Section 2 reviews relevant work, encompassing deep learning techniques applied to 3D point clouds as well as prior studies on the use of spatial autocorrelation features in geospatial object detection for remotely sensed data. Section 3 elaborates on our methodologies, particularly highlighting our novel framework for extracting spatial autocorrelation features from 3D data. Section 4 introduces the datasets used in our study, while Section 5 details two experiments designed to explore the role of spatial autocorrelation in 3D deep learning. The results and findings of these experiments are reported and discussed in Section 6. We draw the conclusions in Section 7.

3.2 Related Work

3.2.1 <u>Related work of 3D deep learning on point cloud data</u>

3D point clouds are inherently unstructured, unordered, and unevenly distributed in space. Handling their unstructured nature is an important problem in 3D deep learning. Early approaches attempted to convert 3D point clouds into structured 3D voxels so that it can be fed to deep neural networks. However, this approach posed challenges as it scaled up in terms of data size (Maturana and Scherer 2015, Qi et al. 2016). One revolutionary work in this domain was the development of PointNet (Qi et al. 2017a), which marked a breakthrough by directly taking unstructured point cloud data as input. PointNet introduced the concept of extracting permutation- and rotation-invariant features (a.k.a., symmetric features) from these unstructured and unordered point clouds. This was achieved through the ingenious use of symmetric functions, specifically max pooling, to aggregate point-wise features to global features that are permutation and rotation invariant. PointNet not only provided architecture for 3D deep learning but also shifted the way 3D deep learning architectures were designed. A generic framework for 3D deep learning based on the empirical knowledge by PointNet (see Figure 3.1) comprises two main components, data sampling and structuralization, and the deep learning architecture.

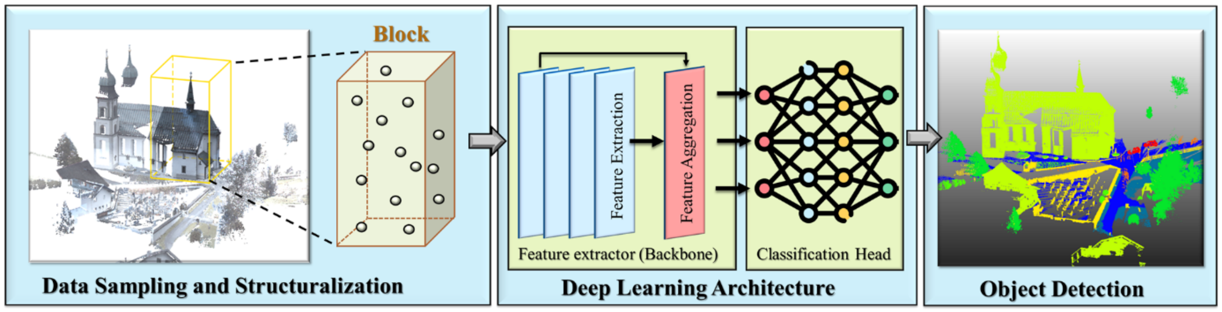

Block
Data Sampling and Structuralization
Feature Extraction
Feature Aggregation
Feature extractor (Backbone)
Classification Head
Deep Learning Architecture
Object Detection

Figure 3.1. Generic framework for 3D deep learning-based object detection. The demonstrated point cloud is from the Semantic3D benchmark dataset by Hackel et al. (2017).

Data sampling involves the spatial partitioning of the input 3D point cloud into smaller and manageable subsets and sampling a fixed number of points from them to make the input structuralized. These subsets are often referred to as blocks (see in Figure 3.1). Data sampling serves two primary purposes, structuring the input and efficient processing. By dividing the point cloud into smaller blocks and sampling with a fixed number of points, input data is structured so that it can be fed into deep learning architectures. On the other hand, data sampling can make the processing more efficient. Working with smaller blocks enables efficient processing, as it reduces computation time and required computing resources. This is crucial for handling large point cloud datasets.

Deep learning architecture learns from structured blocks of points and makes predictions to detect geospatial objects within them. The architecture of deep neural network typically consists of two components, a feature extractor (i.e., backbone) and a classification head. The primary function of a backbone is to extract and aggregate features for further analysis or task-specific applications (e.g., object detection, and classification). The initial layers of the architecture perform feature extraction. After feature extraction, there are neural network layers that aggregate information from neighboring points or an entire block (e.g., a max pooling layer in PointNet), which is crucial for capturing global feature and understanding the relationships between points. There are three main types of modules (Guo et al. 2021) serving as the feature extraction modules including multi-layer perceptron (MLP) layer, convolutional layer, or graph-based layers. PointNet-like methods feed pointwise local features to shared MLP and generate pooling-derived global features (Qi et al. 2017b). CNN-based methods use loosely connected convolutional kernels to extract such invariant features by cascaded layers (Wu, Qi and Fuxin 2019, Boulch 2020). Graph-based methods treat each point as a vertex of a graph to generate invariant features based on neighbors defined by directed edges (Simonovsky and Komodakis 2017, Landrieu and Simonovsky 2018). Classification head refers to the final layers of the architecture (e.g., fully connected layers in a CNN), taking extracted features for predictions.

3.2.2 Spatial autocorrelation in object detection

Many studies prove the effectiveness of spatial autocorrelation features in object detection by explicitly involving such features in machine learning models. The ability of local context awareness of spatial autocorrelation (e.g., semivariogram) is commonly used for representing texture features (Humeau-Heurtier 2019). The incorporation of spatial autocorrelation features is backed by not only practical evidence as aforementioned but also theoretical foundations. The theoretical underpinning of its effectiveness in pattern recognition as explained by Haralick, Shanmugam and Dinstein (1973) is that human beings interpret pictorial information based on spectral, textural, and contextual features, which are the three essential pattern elements of imagery. Spectral information depicts tonal variations in different bands of a spectrum. For example, color information of an image are such values captured by corresponding sensors. Textural information describes the spatial pattern of a spectrum channel. Tso and Olsen (2004) defined it as a joint tonal variation within a prescribed area. Practically, textural features are derived from values within a predefined window from an image, describing the spatial relations between the center point and its neighbors. Contextual features encompass information extracted from a specific section of an image and its surrounding environment, assisting humans to interpret imagery.

Shekhar et al. (2002) explored the source of spatial dependency within remotely sensed data, attributing it to the difference between the fine spatial resolution of the data collected by sensors and the size of the object represented in the data. To illustrate, consider an image of a conference room where a chair is represented by multiple pixels. In this case, the spatial resolution of the imagery is finer than the chair represented in the image. This pixel correlation representing the chair allows us to distinguish it from the background, where the background pixels are less correlated with those belonging to the chair. It is also mentioned as internal spatial continuity (e.g., the chair an object itself) and external spatial discontinuity (e.g., chair and background) by Bian and Xie (2004). Therefore, spatial autocorrelation is inherent in the data collected by sensors (e.g., digital camera and LiDAR), where pixels within a close distance tend to be more similar than distant ones in terms of radiation (Karasiak et al. 2022). As a result, early researchers (Atkinson and Lewis 2000, Haack et al. 2000, Miranda, Fonseca and Carr 1998, Bian and Xie 2004) further fed spatial autocorrelation features (e.g., semivariance) as a representation of texture information to their object detection models.

In the task of object detection from landcover data, it has been observed that different land cover categories exhibit distinct semivariogram patterns (Miranda et al. 1998). The parameters (e.g., range, sill, nugget) estimated for semivariogram models have the potential to be used as the representation of spatial autocorrelation features (Durrieu and Nelson 2013, Pereira et al. 2019). Using semivariograms as additional features to geospatial data feeding to machine learning models have been suggested with advantages including effectiveness in increasing the accuracy of object detection (Kattenborn et al. 2021). Other than parameters of a semivariogram model, semivariance is also used as a spatial autocorrelation feature. The studies incorporating semivariance for object detection showed significant improvement in model performance (Miranda et al. 1998, Miranda, Macdonald and Carr 1992, Zawadzki et al. 2005, Bian and Lee 2005, Bian and Xie 2004). In terms of the improvement in previous geospatial object detection tasks brought by including spatial autocorrelation features, this study investigated the potential of its capability in a 3D context.

3.3 Methodology

3.3.1 3D Deep Learning Framework

In this section, we present our 3D deep learning framework for the explicit incorporation of spatial autocorrelation to detect geospatial objects (see Figure 3.2). The input data is, for example, point cloud with spatial and color information that can be retrieved via LiDAR techniques. The spatial autocorrelation feature is extracted based on both information through the approach to be discussed in Section 3.3.2. The spatial autocorrelation features are subsequently concatenated with the spatial and color information – serving as the input layer of 3D deep learning. Through training and validation, the model is equipped with prediction capabilities to detect geospatial objects (e.g., trees, grass, and road as demonstrated in Figure 3.2).

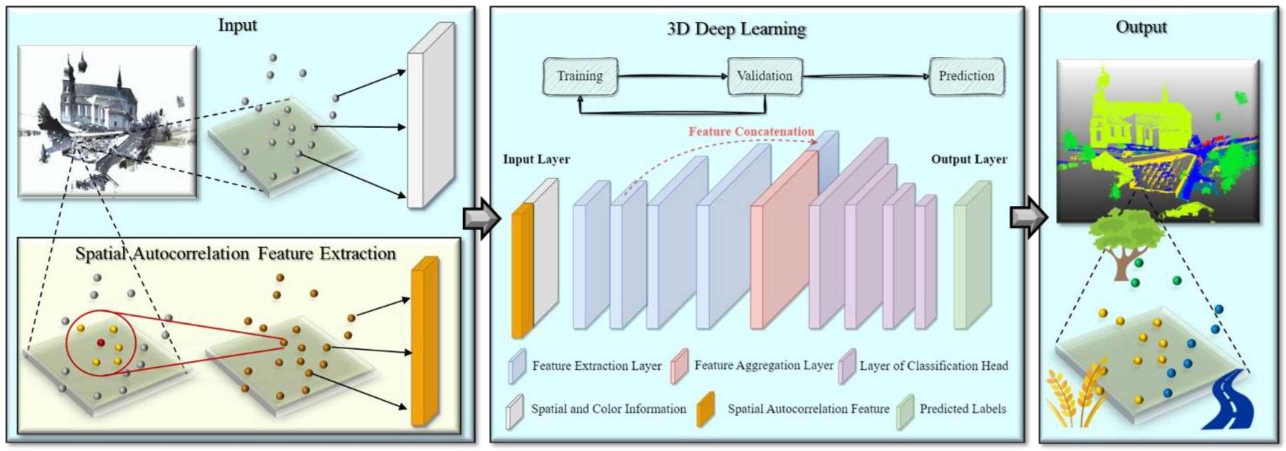

Input
Spatial Autocorrelation Feature Extraction
3D Deep Learning
Training
Validation
Prediction
Feature Concatenation
Input Layer
Output Layer
Feature Extraction Layer
Feature Aggregation Layer
Layer of Classification Head
Spatial and Color Information
Spatial Autocorrelation Feature
Predicted Labels
Output

Figure 3.2. 3D deep learning framework with explicit incorporation of spatial autocorrelation for geospatial object detection. A typical 3D deep learning architecture is used for demonstration purpose. The demo point cloud is from the Semantic3D benchmark dataset.

In this study, we used PointNet as a representative 3D deep learning architecture. First, PointNet is highly efficient in terms of computation due to its relatively low number of parameters favored by shared Multilayer Perceptron (MLP) layers, which allows for rapid training and validation. The clean and straightforward design of the architecture not only accelerates the training process but also improves the interpretability of the deep learning model. Second, its concise architecture offers us direct control over hyper-parameter configurations, enabling us to carefully assess how various features affect the model’s ability to detect geospatial objects. Third, its strong representational power, as highlighted in the study by Guo et al. (2021), as another reason to support our selection. Finally, its capability to effectively capture spatial hierarchies in 3D data renders it outstanding for object detection tasks. Given these considerations, PointNet stands out as an ideal candidate for this study – It offers a balanced combination of efficiency, interpretability, and representational power, thus making it well-suited for our study.

PointNet was not originally implemented with a primary focus on large-scale semantic segmentation. Instead, it is designed for general purposes, such as classification, part segmentation, and semantic segmentation (Qi et al. 2017a). Therefore, we modified the PointNet for large-scale semantic segmentation in our study by excluding the T-net (spatial transformer network) module (Jaderberg, Simonyan and Zisserman 2015), which was designed to automatically rotate point clouds making small-scale objects spatially invariant. For small-scale semantic segmentation (i.e., part segmentation on point cloud of an object), T-net can help make the input spatially invariant no matter how it was spatially transformed. However, it is not needed for large-scale semantic segmentation (i.e., scene parsing) since the x-y plane will always be on ground and the z-axis is the only axis on which rotation can be operated. Therefore, for large-scale semantic segmentation, it is simply required to make the z-axis rotation invariant. The design of PointNet++ for scene parsing can support this point, which excluded the T-net module. The modified architecture of PointNet is shown in Figure A1 and we implemented it using PyTorch (version 1.12.1).

3.3.2 Retrieve spatial autocorrelation features in 3D point cloud

In this section, we introduced the extraction of spatial autocorrelation features (specifically the semivariance) from unstructured 3D point clouds. Semivariance, essentially an experimental semivariogram, enables us to measure the spatial decay of a variable over distance (referred to as lag) by calculating and aggregating pairwise differences of observed value (specifically color information in our case) between any locations within a specified extent.

Early approaches, as pioneered by Miranda et al. (1992), Miranda and Carr (1994), Miranda et al. (1998), Bian and Lee (2005), Bian and Xie (2004), Kamal, Phinn and Johansen (2014), and Wu et al. (2015), directly used semivariance at different lag values as textural features, or spatial continuity and discontinuity properties for object detection on remotely sensed data. Typically, in the 2D context, a fixed-size moving window ($r*r$; r: window size) was used to extract a subset of pixels centered around each pixel, where semivariance is calculated at each spatial lag within the window. However, when dealing with the inherently unstructured points within a 3D point cloud, the potential absence of points within a predefined spatial lag presents a unique challenge, resulting in a null semivariance value for the lag bin. Therefore, it poses a challenge to derive semivariance from a 3D point cloud for use in geospatial object detection.

We propose a novel approach to address this challenge. Our solution, detailed in Figure 3.3, introduces a four-step approach to tackle the aforementioned challenge, namely: searching for neighbors, estimating pairwise differences, binning, and estimating semivariance. Each of these steps is elucidated further in the following subsections. Furthermore, we provided more insights about applicability of the traditional approach to extract semivariance for point cloud data.

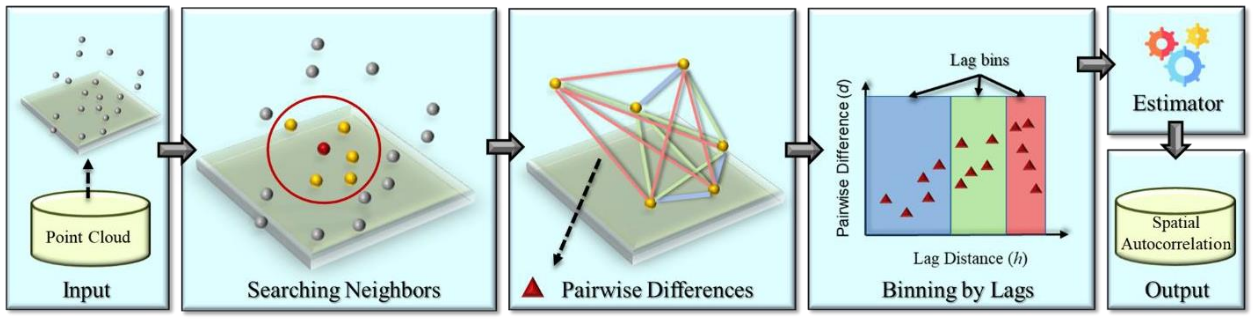


Figure 3.3. Spatial autocorrelation feature extraction – a demo of deriving semivariance for one point.

3.3.2.1 Searching neighborhood

A neighborhood is a band of distance or number of points that all points within the neighborhood will be used for deriving the semivariance. In traditional studies, a neighborhood is defined by a certain square window. For example, 22*22 window size and 7*7 kernel size are used by Miranda and Carr (1994) for their study since this particular size is identified by the trade-off between semivariance estimation accuracy and lowering the risk of the moving window overlapping on class boundary. However, a certain distance may not be suitable for an unevenly distributed point cloud, where there might not always be a neighbor within a certain neighborhood (a certain radius for neighbor searching is demonstrated in Figure 3.4A). To make it more adaptive to the irregularly spaced nature of point clouds, we use k-nearest neighbor (kNN) as the neighborhood searching method as shown in Figure 3.4B. Therefore, there will always be neighbors in a neighborhood.

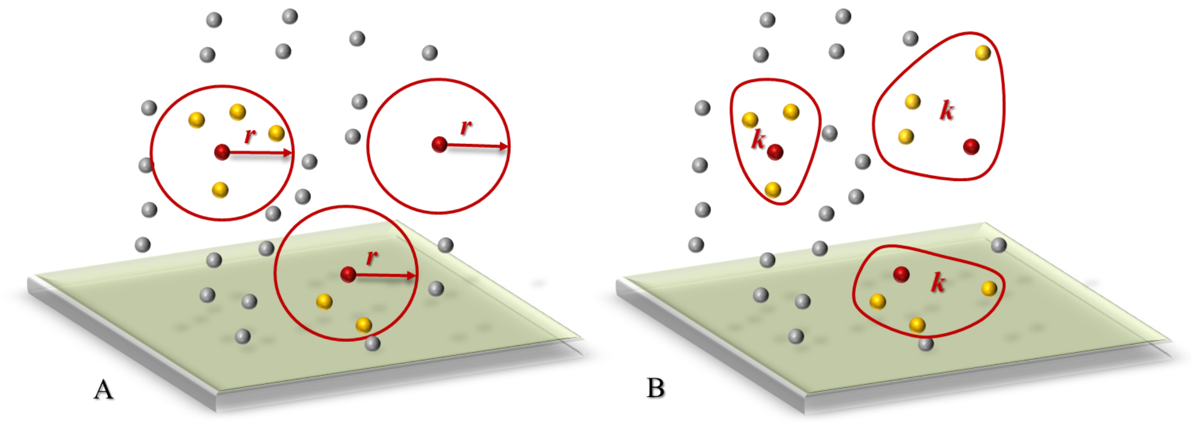


Figure 3.4. Conceptual illustration of neighborhood searching methods. Red dots are points of interest, where the searching is conducted. Yellow dots are the neighbors within radius *r* or *k* nearest neighbors with respect to the selected point. Gray dots are points outside of the neighborhood ranges. A: searching with a defined radius *r* in Euclidean space; B: searching with k-nearest neighbor (e.g., *k* = 3 as shown).

3.3.2.2 Binning

The method of grouping pairwise differences into categories is termed “binning”. Traditional binning is to classify by equal intervals, where each lag bin encompasses a uniform range of lag distances (as illustrated in Figure 3.5A). However, this approach can be problematic for 3D point clouds due to their varied density distribution. Specifically, equal intervals may result in null values in empty bins, introducing issues to subsequent deep learning. To address this challenge, we adopt quantile classification, which allocates pairwise differences to lag bins based on equal number of pairs. This ensures a more uniform distribution of pairwise differences across lag bins (as shown in Figure 3.5B), eliminating the risk of null values in any lag bin.

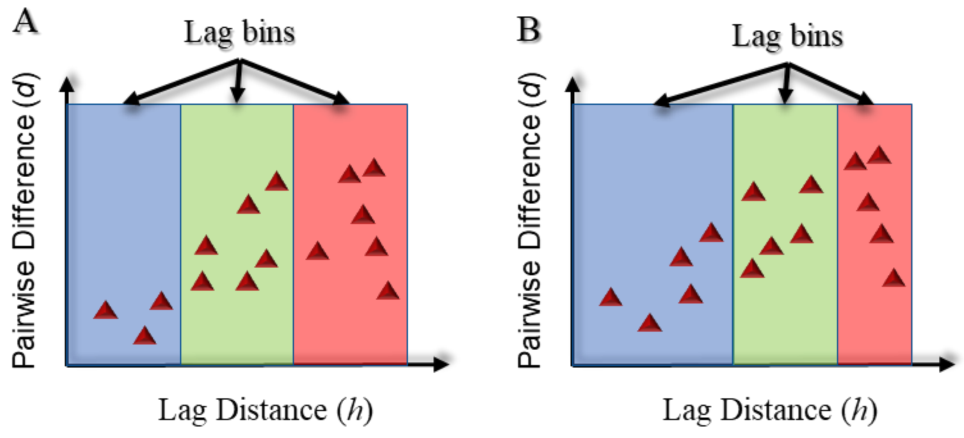


Figure 3.5. Demonstration of binning methods. A: equal interval; B: quantile. There can be scenarios where there are no pairwise difference values falling into some lag bins classified by equal interval.

The combination of the kNN searching and quantile binning is the key of the proposed approach, which can address the challenge brought by the unstructured nature of 3D point cloud as aforementioned. We also observe that this solution might bring uncertainty to the scale for semivariance calculation as the spatial extent defined by kNN can be different across points. This impact on model generalization can be mitigated by the sampling strategies used in this study. For example, the training samples are randomly generated by a fixed size of blocks. Moreover, the points within a block are also randomly sampled (see detailed configurations in Section 3.4). That is, for the same point, it can be captured multiple times with different nearest neighbors. This idea was also used in Klemmer, Safir and Neill (2023), who used such random sampling before convoluting k nearest neighbor to mitigate the impact of the scale so as that to learn scale-insensitive features.

3.3.2.3 Estimator of semivariance

Various estimators can be employed to compute experimental semivariance, including but not limited to methods developed by Matheron (1963), Dowd (1984), and Cressie and Hawkins (1980). In our study, we focus on utilizing the two foundational semivariance estimators from Matheron (1963) and Dowd (1984), which inherently rely on the mean and the median of pairwise differences. The Matheron estimator's formula for calculating semivariance at a given spatial lag *h* is presented in Equations 3.1 and 3.2:

$$\gamma(h) = \frac{1}{2N(h)} * \sum_{i=1}^{N(h)} (d)^2 \tag{3.1}$$

$$d = Z(x_i) - Z(x_{i+h}) \tag{3.2}$$

where *Z(x)* is the observation (e.g., color information) at the location x. *N* is the number of point pairs at the bin lag *h*; *d* is the pairwise differences at a given lag *h*.

The Matheron estimator computes the mean of the squared pairwise differences as semivariance. However, Matheron semivariance is sensitive to extreme values by using means. The Dowd estimator uses the median value, which is robust to extreme values. The formula of the Dowd estimator is defined in Equations 3.3 and 3.4.

$$\gamma(h) = \frac{1}{2} * median(d)^2 \tag{3.3}$$

$$d = Z(x_i) - Z(x_{i+h}) \tag{3.4}$$

The computational complexity (Cormen et al. 2022), denoted as O, of deriving the semivariance for one point with respect to the number of nearest neighbor, k, is $O(k^2)$ because the number of pairwise differences is $k*(k-1)/2$. Therefore, k is one important hyperparameter to be configured for semivariance derivation considering computing resources. An increasing k can result in dramatic increase in the amount of computation.

3.4 Data

In this study, we leveraged two benchmarks to evaluate the effectiveness of spatial autocorrelation features in diverse scenarios (i.e., indoor and outdoor environment settings) for 3D deep learning. These benchmarks are listed as follows.

1. S3DIS (Stanford 3D Indoor Scene Benchmark): S3DIS (Armeni et al. 2016) is a representative dataset of small-scale indoor scenes. It serves as an essential resource for understanding indoor environments, including room- and object-level classification.
2. Semantic3D (a.k.a Large-scale Point Cloud Classification Benchmark): Semantic3D (Hackel et al. 2017) provides a large-scale outdoor point cloud classification benchmark. It is a valuable dataset for evaluating algorithms designed for point cloud processing and classification tasks in expansive outdoor environments.

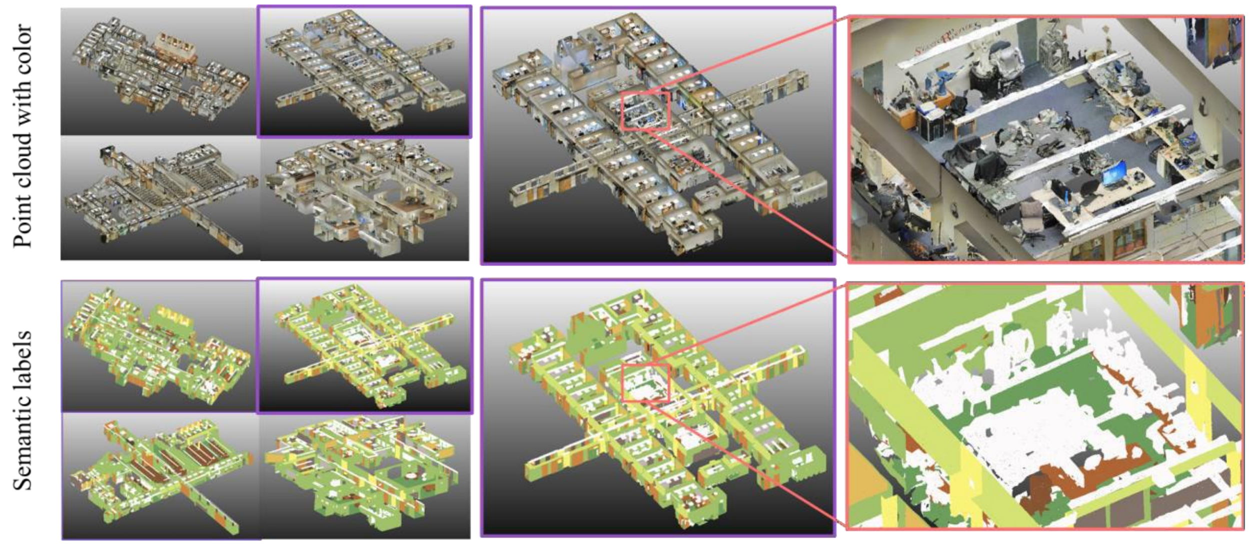
We aim to offer a better understanding of how explicit incorporation of spatial autocorrelation impacts 3D deep learning in different environment settings. Specifically, we sought to determine if semivariance improves object detection in both narrow indoor spaces and expansive outdoor settings.

3.4.1 Stanford 3D Indoor Scene Dataset

S3DIS is a representative indoor scene dataset with 13 semantic classes, including ceiling, floor, wall, beam, column, window, door, table, chair, sofa, bookcase, board, and clutter. There are in total 271 indoor scene of rooms (e.g., office, conference room, classroom, etc.) across 6 main areas, namely Areas 1-6, where each area represents a part of a building as demonstrated in Figure 3.6 (4 of 6 are shown for demonstration purpose). In this study, Area 1-4 and 6 are used for training, and Area 5 is used for validation purpose as per previous studies working on the dataset (Qi et al. 2017b, Fan et al. 2021). The preprocessing generally followed the steps outlined in Qi et al. (2017b), where 4,096 points are sampled from each one meter block.

However, we dropped any blocks with less than 128 unique points to fit downstream processing of semivariance. Based on the current block generating template, the points within a block are resampled (with replacement). If the original number of points is less than 4,096 but the output is oversampled to 4,096, there must be duplicated points. Therefore, we only take unique point location into consideration so that it will not be impacted by duplicated points resulted by the resampling template. In this case, if the number of unique points is less than 128, it will violate one configuration (k = 128) in Experiment 2 (see Section 3.5.2 for detail).

We demonstrated the semivariance value for each class along the bins in Figure 3.7. Generally, we can observe that semivariance increases along the lag bin which is reasonable as the values can differ more in distant points. Moreover, semivariance of different objects show various responses to the lag bins. The patterns of their semivariance values along the lag bins can potentially help identify them. We observed that floor and ceiling had a similar pattern. They both may have less variance of texture within the given lag bin.



Ceiling ▨ Floor 🟩 Wall 🟩 Beam 🟨 Column 🟨 Window 🟧
Door 🟧 Table 🟫 Chair 🟫 Sofa 🟫 Bookcase 🟫 Board ⬜ Clutter ☐

Figure 3.6. Demonstration of S3DIS datasets. The ceiling is hidden in this figure so that the objects within each room can be visualized.

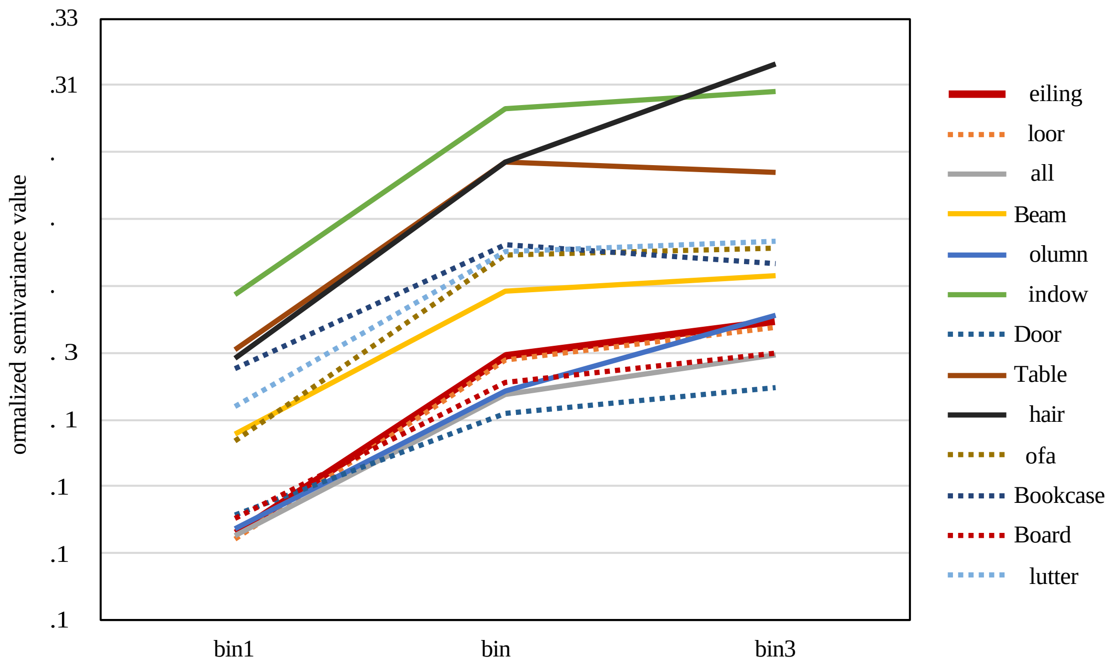


Figure 3.7. Demonstration of averaged semivariance value derived for each class in S3DIS dataset. The semivariances are derived by Matheron estimator with 16 nearest neighbors for demonstration purpose.

3.4.2 Large-scale point cloud classification benchmark

Semantic3D (Hackel et al. 2017) is a large-scale outdoor scene dataset with 8 semantic classes including man-made and natural terrain, high and low vegetation, building, hard scape, scanning artefacts, and cars (see Figure 3.8). There are 15 labeled rural and urban scenes. We used 9 of them for training and the rest for validation. This outdoor scene dataset is challenging in 3D deep learning due to its large spatial extent and the extremely uneven distribution in the space.

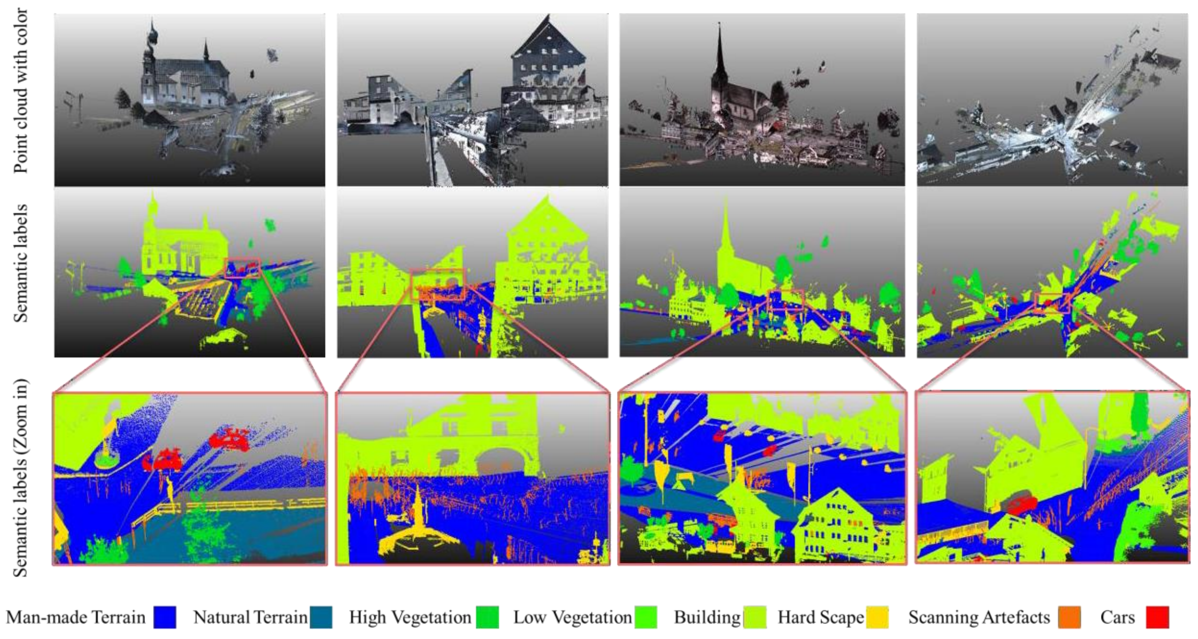


Figure 3.8. Demonstration of the Semantic3D dataset. Semantic3D dataset has 15 labeled scenes ranging from rural to urban areas with 8 classes.

We preprocessed the dataset by using 8 meters as the block size as suggested by previous study (Boulch 2020) and 4,096 points as the number of points. The semivariance values are calculated prior to being fed into the 3D deep learning model. We demonstrate an averaged value for each class in Figure 3.9. Car has the highest value as car since the texture of a car is more complex given a defined neighborhood (i.e., k=16 in this case). Natural terrain and high vegetation show a lower value indicating the color of points are more similar within the given bin.

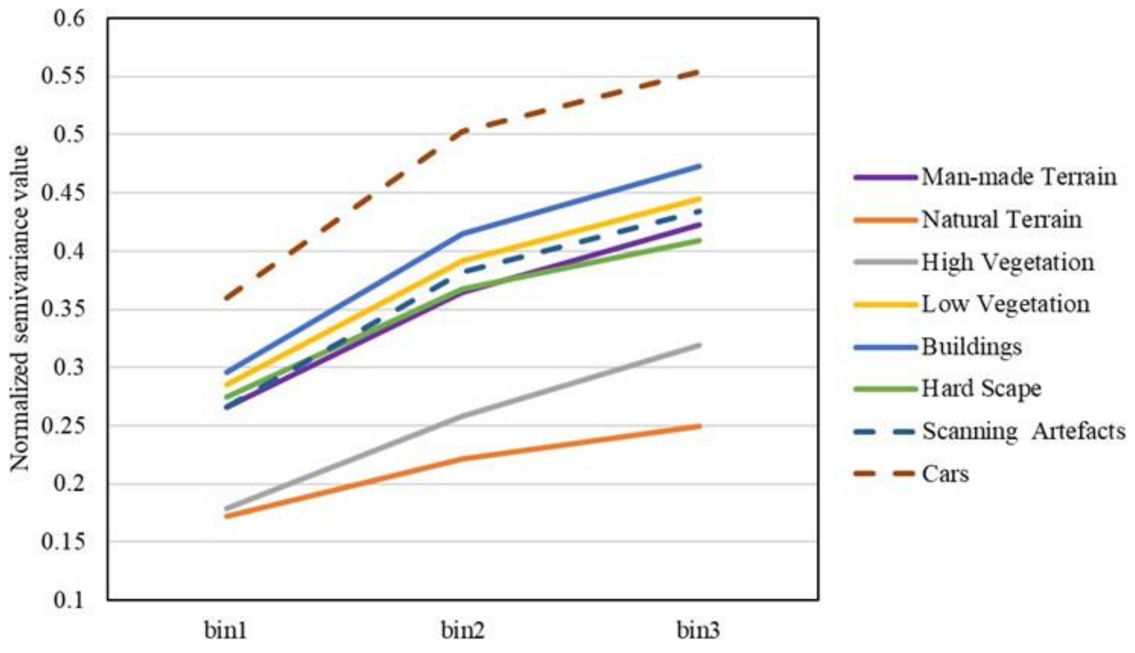


Figure 3.9. Demonstration of averaged semivariance value derived for each class in Semantic3D dataset. The semivariances are derived by Matheron estimator with 16 nearest neighbors for demonstration purpose.

3.5 Experimental Design

In this study, we designed two experiments to address the research question of how the explicit incorporation of spatial autocorrelation features impacts 3D deep learning on object detection in different environment settings. The first one investigates whether semivariance is beneficial to the model performance. The other one perturbs the configuration of semivariance generation and explores insights of different configurations towards various environments.

3.5.1 Experiment 1: Effectiveness of spatial autocorrelation features in 3D deep learning

This experiment aims to assess the impact of explicitly incorporating semivariance as a spatial contextual feature on the performance of 3D deep learning models. Specifically, the experiment compares models trained solely on spatial information against those trained on a combination of spatial, color, and/or semivariance data.

We designed three treatments with different features in input data (see Table 3.1), spatial information only, color information as additional features, and spatial autocorrelation features as the other additional feature channels. The semivariance of 3 lag bins – 3 additional feature channels – are generated based on 16 nearest neighbors within a block of points, where the configurations of blocks for different datasets refers to Sections 3.4.1 and 3.4.2. To address the uncertainties introduced by the randomization in the deep learning process, we trained and validated the model for each treatment with 10 repetitions. The number of repetitions was influenced by the computational challenges against our available computing resources. Additionally, our results (see Appendix 3.2 for detail) suggest that 10 repetitions were adequate for reliable outcomes. The validation dataset was pre-generated so that the performance measurements were comparable. Then, we compared the averaged performance measurements over the 10 repetitions, where Intersection over Union (IoU), mean Intersection over Union (mIoU) and Overall Accuracy (OA) were used as indicators to identify how well the model generalizes on a validation dataset. IoU and OA are two measurements commonly used for object detection (Rezatofighi et al. 2019, Qi et al. 2017a, Boulch 2020). Jaccard similarity index or Lee-Salle shape index (Lee and Sallee 1970), were commonly used in the domain of geography or GIScience (Shelton 2019, Clarke 1996), which have a similar concept of IoU. The equations of the three metrics are shown below.

$$IoU = \frac{TP_i}{TP_i + FN_i + FP_i} \tag{3.5}$$

$$mIoU = mean(IoU_i) \tag{3.6}$$

$$OA = \frac{TP_{all}}{TP_{all} + FP_{all}} \tag{3.7}$$

where *TP*, *FP*, and *FN* are True Positive, False Positive, and False Negative; subscript *all* means across all classes and *i* refers to each individual class.

Table 3.1. Configuration of Experiment 1.

Treatment ID	Features	Attributes	Repetition
1	Spatial information only	XYZ	10
2	Spatial and color information	XYZ + RGB	10
3	Spatial, color, and spatial autocorrelation information	XYZ + RGB + Semivariance	10

After training, the average performance metrics across the 10 repetitions were calculated for each treatment. Subsequently, comparisons were conducted to assess the observed differences. By employing the systematic approach, this experiment aims to offer an in-depth understanding of the effectiveness of semivariance as a spatial autocorrelation feature in 3D deep learning models for geospatial object detection.

3.5.2 Experiment 2: Investigation into Uncertainty of Spatial Context affected by Configurations

The objective of this second experiment is to investigate the uncertainty of contextual information with respect to the effectiveness of spatial autocorrelation features that are affected by the configurations. The parameters perturbed for the configuration are estimators (i.e., Matheron estimator and Dowd estimator) and the number of nearest neighbors (*k*), which specifies the size of the local neighborhood used for computing semivariance as shown in Table 3.2.

The challenges of *k* value settings come from two aspects. A smaller value for *k* may result in loss of color variation within the local neighborhood, particularly when the LiDAR sensor offers a finer resolution than the RGB color sensor at a certain distance[7]. A larger *k* value poses computational challenges, as the computational complexity (see detail in Section 3.3.2.3) scales quadratically with $O(k^2)$. Through initial exploratory experiments and trial runs, we have identified a *k*-value range of 8 to 128 as suitable for further investigation. These values were chosen based on their balance between capturing local variations and maintaining computational efficiency.

Multiple datasets with varying characteristics were used to train and validate the models. For each combination of *k* and estimator, we trained and validated the deep learning models. Each configured model was trained 10 times to investigate uncertainties inherent in deep learning processes brought by the stochastic processes, for example, the random initialization of weights. The training data and validation data were pre-generated so as to avoid extra time spent on generating the data and make the outcomes comparable, where 3 bins of semivariance values were derived for each point with respect to a number of nearest neighbors within a block of points. We computed mean and variance for the model's performance metrics across all repetitions. Metrics such as IoU for each class, mIoU of classes, and OA serve as key indicators for assessing the models' performance.

By systematically manipulating these parameters and analyzing the performance outcomes, this experiment seeks to provide insights to configurations for utilizing semivariance in object detection from different scenarios (i.e., indoor and outdoor scenes).

[7] The spatial information and color information of point clouds are fused based on laser scanner and RGB sensor (e.g., camera). Due to their different resolutions, points can correspond to the same pixel from RGB imagery.

Table 3.2. Configuration of Experiment 2.

Treatment	***k***	**Estimator**	**#Repetition**
1	8	Matheron	10
2	16	Matheron	10
3	32	Matheron	10
4	64	Matheron	10
5	128	Matheron	10
6	8	Dowd	10
7	16	Dowd	10
8	32	Dowd	10
9	64	Dowd	10
10	128	Dowd	10

**k* refers to *k* nearest neighbors.

3.6 Results and Discussion

3.6.1 Computing Performance of the Experiments

To overcome the computational challenge brought by repetitions of model training for each treatment in this study, we leverage high-performance computing (HPC) cluster resources. HPC has been widely utilized to assist geospatial research in addressing computational challenges (Tang and Wang 2020). Furthermore, HPC equipped with Graphics Processing Units (GPUs) enables unprecedented power to solve geospatial problems (Tang and Jia 2014). Specifically in this study, we have 10 nodes with GPUs in our HPC cluster, where each node has 1 NVIDIA A40 GPU, 8 cores of an Intel Xeon Gold 6326 CPU, and 64 GB memory. The total sequential computing time for Experiments 1 and 2 are about 368.07 hours and 1,362.68 hours. The computing time by leveraging 10 nodes in parallel for the two experiments are 38.15 hours and 142.78 hours. They are sped up by 9.65 and 9.54 times. The average computing time and the corresponding standard deviation for each treatment for corresponding datasets are shown in Table 3.3. As a result, GPU-based HPC computing resources provide solid support for the computational needs of training of our 3D deep learning model as required by the experiments in this study.

Table 3.3. Computing time for each type of treatment categorized by features included (Std: standard deviation).

Treatment	**Dataset**	**Number of Channel**	**Average Computing Time**	**Std. Computing Time**
Spatial information only	Semantic3D	3	8.56 hours	0.35 hours
Spatial and color information	Semantic3D	6	8.95 hours	0.18 hours
Spatial, color, and spatial autocorrelation information	Semantic3D	9	10.35 hours	0.28 hours
Spatial information only	S3DIS	3	2.92 hours	<0.01 hours
Spatial and color information	S3DIS	6	2.99 hours	<0.01 hours
Spatial, color, and spatial autocorrelation information	S3DIS	9	3.02 hours	0.01 hours

3.6.2 Effectiveness of Spatial Autocorrelation Features in 3D Deep Learning

This underscores the integral role that color information plays in providing essential context for geospatial object detection in 3D deep learning. We report the results on the two benchmark datasets, S3DIS and Semantic3D in the following subsections.

3.6.2.1 Results on Semantic3D dataset

Table 3.4 summarizes the comparison among the three treatments for Semantic3D dataset while using Treatment 2 (i.e., the one with spatial and color information) as the baseline. The detailed results of 10 repetition treatments for each treatment are presented in Appendix 3.2 (see Tables A1-A4). Observed from Table 4, the baseline treatment has averaged OA and mIoU values of 81.95% and 51.58%, respectively. However, omitting color information from the dataset led to substantial decrease in performance metrics. Specifically, OA declined by 15.46%, and mIoU dropped by 20.84% (Table 4). Thus, incorporating semivariance as an additional feature provided a moderate but meaningful increase in performance: OA improved by 1.37%, and mIoU saw a 2.47% increase.

As evident in the IoU metrics for specific classes, the inclusion of semivariance not only enhances global accuracy but also boosts performance at the class level. The IoU across classes exhibited gains between 0.33% and 5.17%. Notably, the IoU scores for scanning artefacts and cars saw significant improvement, around 5%.

Table 3.4. Performance comparison across treatments (Semantic3D Dataset).

Measurements	**Treatment 1**	**Treatment 2 (Baseline)**	**Treatment 3**
OA	66.49% (-15.46%)	81.95%	83.32% (+1.37%)
mIoU	30.74% (-20.84%)	51.58%	54.05% (+2.47%)
Man-made terrain	69.18% (-21.51%)	90.69%	91.02% (+0.33%)
Natural terrain	27.03% (-45.56%)	72.59%	73.45% (+0.86%)
High vegetation	36.80% (-20.21%)	57.01%	60.15% (+3.14%)
Low vegetation	8.58% (-15.54%)	24.12%	26.41% (+2.29%)
Buildings	65.64% (-14.21%)	79.85%	81.38% (+1.53%)
Hard scape	11.80% (-9.75%)	21.55%	23.19% (+1.64%)
Scanning artefacts	12.60% (-7.65%)	20.25%	25.10% (+4.85%)
Cars	14.29% (-32.26%)	46.55%	51.72% (+5.17%)

*OA: Overall Accuracy. mIoU: mean Intersection over Union. The class name (e.g., cars) indicates the IoU for each class. The values are averaged across the repetitions.

These results suggest that feature selection plays a crucial role in the performance of 3D deep learning models on semantic datasets. Specifically, the incorporation of color information and spatial autocorrelation features can substantially enhance model generalization capabilities in terms of the performance metrics.

3.6.2.2 Results on S3DIS benchmark

Table 3.5 summarizes the comparison between three treatments for the S3DIS dataset, using the treatment with both spatial and color information as the baseline. Appendix 3.2 reports detailed results from 10 repetitions for each treatment (see Tables A5-A8).

In terms of global accuracy, the baseline treatment exhibits average OA and mIoU values of 82.83% and 50.19% (see Table 5). Omitting color information led to a substantial drop in these metrics: OA declined by 3.03%, and mIoU fell by 4.88%. In contrast, the incorporation of semivariance as an extra feature resulted in moderate improvements. Specifically, OA increased by roughly 1.36%, and mIoU improved by 2.35%.

Regarding IoU for 13 classes in S3DIS data, the treatment 3 outperformed other treatments in predicting most of the classes. According to the IoU values for individual classes, the average IoU across different classes ranged from 0.04% to 5.44%. In particular, the IoUs for board and bookcase saw a remarkable rise of around 4-5%.

The treatment with spatial information only shows dramatical weakness in understanding windows and doors with 17.95% and 20.92% lower than those of the baseline treatment. This can be due to the spatial structure of window and door that is very close to that of a flat wall so that models trained on spatial information-only dataset appear to be less accurate in this scenario. It is noteworthy that the simulation shows the highest IoU for the sofa and column class among the three treatments. Boulch (2020) also found that the column class is better detected by a no-color model than the color model. Boulch attributed this to more importance given to RGB features in the presence of color information during the stochastic training process, but columns often share the same color. Therefore, color does not serve as a distinguishing feature, resulting in a worse performance for such classes (Boulch 2020). Similarly, we found that both chair and sofa exhibited comparable behaviors.

In summary, color information can help improve the performance of the model trained only on spatial information in our experiments. Engelmann et al. (2017) also advocated this finding in their experiments that there was approximately 3% improvement in OA and mIoU when color was added. Our experiments further suggest that semivariance, representative of spatial autocorrelation features, can boost the model performance of 3D deep learning. This aligns with the findings by Bian and Xie (2004) that semivariance can help identify geospatial objects but our study focuses its capability in 3D context. However, the uncertainty introduced by this framework for deriving semivariance, as opposed to the conventional method of searching neighbors within a given radius and binning with fixed lag distances, is worth further investigation in future work. This need arises especially due to the spatial heterogeneity in point density, which may affect robustness of spatial autocorrelation features. These features are derived from neighbors identified through the kNN searching process. To illustrate, in areas of sparse point density, some neighbors identified by kNN might not actually be relevant to the reference point. They could be so far apart that spatial dependency might no longer exist. Such scenarios highlight potential limitations in our current approach that need more in-depth analysis. From our understanding, we assume that an adequate selection of block size (see Figure 3.1), for partitioning and sampling the point cloud could help mitigate such effect because that allows the models to be trained on samples with varying point densities, potentially enhancing the robustness of model performance.

Table 3.5. Performance comparison across treatments (S3DIS Dataset).

Measurements	**Treatment 1**	**Treatment 2 (Baseline)**	**Treatment 3**
OA	79.80% (-3.03%)	82.83%	84.19% (+1.36%)

mIoU	45.31% (-4.88%)	50.19%	52.54% (+2.35%)
Ceiling	87.02% (-2.01%)	89.03%	90.82% (+1.79%)
Floor	96.68% (-0.46%)	97.14%	97.18% (+0.04%)
Wall	59.92% (-6.07%)	65.99%	68.54% (+2.55%)
Beam	44.80% (-0.70%)	45.50%	46.90% (+1.40%)
Column	9.82% (+3.83%)	5.99%	7.72% (+1.73%)
Window	19.45% (-17.95%)	37.40%	39.04% (+1.64%)
Door	37.86% (-20.92%)	58.78%	61.64% (+2.86%)
Table	56.88% (-2.37%)	59.25%	60.51% (+1.26%)
Chair	62.99% (+1.96%)	61.03%	64.29% (+3.26%)
Sofa	26.05% (+19.72%)	6.33%	9.00% (+2.67%)
Bookcase	35.13% (-5.07%)	40.20%	44.29% (+4.09%)
Board	9.99% (-28.55%)	38.54%	43.98% (+5.44%)
Clutter	42.40% (-4.95%)	47.35%	49.07% (+1.72%)

*OA: Overall Accuracy. mIoU: mean Intersection over Union. The class name (e.g., sofa) indicates the IoU for each class. The values are averaged across the repetitions.

3.6.2.3 Summary of findings in different scenarios

In terms of 3D deep learning for geospatial object detection, the importance of feature selection cannot be overstated. The performance of model on generalization capabilities, and understanding of specific classes, are heavily influenced by the choice of features. An observation from the experiments on both datasets (see Sections 3.6.2.1 and 3.6.2.2) was the significant impact of color information on performance. Specifically, the omission of color data resulted in a marked reduction in key performance metrics, such as OA and mIoU in different scenarios (indoor and outdoor environments). This underscores the integral role that color information plays in providing essential context for geospatial object detection in 3D deep learning. Moreover, the spatial autocorrelation of color information, specifically semivariance, further produces moderate but significant improvements in model performance across both datasets. This is evident not only in global metrics such as OA and mIoU but also in class-specific IoU metrics. The ability of semivariance in capturing spatial relationships and providing local contextual information was particularly pronounced, with certain classes showing substantial performance boosts (e.g., 5%

improvement in IoU of cars in Semantic3D dataset). The experiments further reveal the advantages of integrating spatial autocorrelation features, with semivariance to boost model performance.

3.6.3 Uncertainty of spatial context

Experiment 2 investigates the uncertainty of spatial context brought by the spatial autocorrelation features (i.e., semivariance) affected by diverse configurations (e.g., number of nearest neighbors). To do this, we identified the optimal settings for generating effective semivariance metrics for 3D object detection across various scenarios. We perturbed the number of nearest neighbors, *k*, from 8 to 128 using a quadratic increase step and compared the performance of two semivariance estimators: Matheron estimator and Dowd estimator. In this section, we present the results for treatments that involved perturbed *k* with the Matheron estimator, as the outcomes with the Dowd estimator exhibited similar patterns. Tables A11 and A12, and Figures A2 and A3 from Appendix 3.4 report the detailed results of the Dowd estimator.

3.6.3.1 Number of Nearest Neighbors for Outdoor Environment

Table 3.6 depicts the mean values of performance metrics for each configuration. The OA and mIoU values range from 82.80% to 83.14% and 52.81% to 53.52%, respectively. These results suggest that the overall performance is relatively unaffected by variations of *k*. A moderate decline in the mean values of OA and mIoU was noted with an increasing *k*, though there was an exception at *k* = 64. We conducted one-tailed t-test to explore if they have significant difference, see Table A9. The t-test results suggest that only the results of k=8 appear to only be significantly higher than that of k=32 (both OA and mIoU) and k=128 (mIoU only) at a 90% confidence level. Regarding the OA and mIoU, a smaller *k* generally results in a higher IoU in the outdoor environment.

Table 3.6. Performance metrics across different k values (Semantic3D dataset).

Measurements	***k* = 8**	***k* = 16**	***k* = 32**	***k* = 64**	***k* = 128**
OA	83.14%	82.87%	82.78%	82.97%	82.80%
mIoU	53.52%	53.19%	53.00%	53.34%	52.81%
Man-made terrain	91.08%	90.57%	90.83%	90.89%	90.71%
Natural terrain	73.93%	72.53%	73.43%	73.16%	72.54%
High vegetation	60.17%	59.72%	58.93%	59.47%	59.44%
Low vegetation	25.70%	24.95%	24.67%	25.25%	25.14%
Buildings	81.53%	81.18%	81.00%	81.03%	81.13%
Hard scape	22.37%	22.89%	22.95%	22.99%	22.58%
Scanning artefacts	24.02%	24.22%	22.93%	23.79%	23.39%
Cars	49.39%	49.44%	49.29%	50.14%	47.56%

**k*: number of nearest neighbors. OA: Overall Accuracy. mIoU: mean Intersection over Union. The class name (e.g., Cars) indicates the IoU for each class. The values are averaged across the repetitions.

Figure 3.10 shows the variation across ten repetitions for each configuration. Generally, the mIoU and OA for the ten repetitions varied between 49.5% to 55.5% and 81.0% to 84.5%, respectively. A noticeable trend suggests a slight decrease in accuracy with a rising *k*. In summary, a smaller *k* is preferred in an outdoor environment. Specifically, the treatment with *k* = 8 exhibits the best performance compared to others.

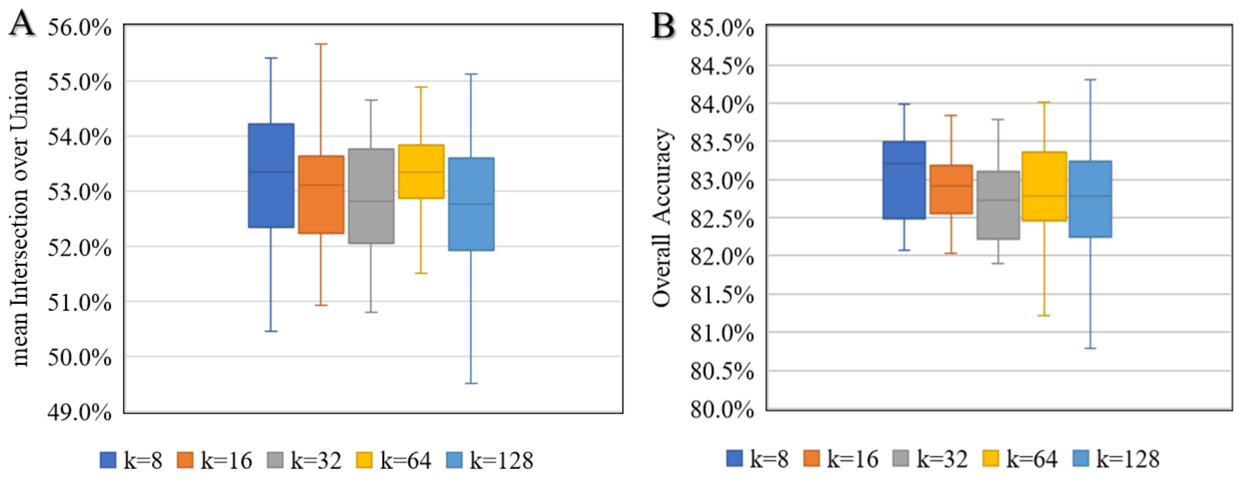


Figure 3.10. Box chart of performance measures for the Semantic3D dataset with Matheron semivariance. A: mean Intersection over Union; B: Overall Accuracy. Centerline is for median.

3.6.3.2 Number of Nearest Neighbors for Indoor Environment

The performance metrics showcased in Table 3.7 reveal patterns for the S3DIS dataset. The metrics exhibit consistent trends across different *k*. Specifically, the mIoU values fall in between approximately 52% to 53%. Similarly, the OA values span from approximately 84% to 85%. One noticeable trend is the gradual increase in both averaged OA and mIoU as *k* increases. This trend reaches a peak when *k* = 128, with the model achieving 84.56% of OA and 53.04% of mIoU. The results of one-tailed t-test are shown in Table A10. The performance (i.e., both OA and IoU) of *k* = 8 is significantly lower than the other three treatments; moreover, when *k* = 128, the performance is significantly higher than others at a 95% confidence level. The trend is not significant when we compare the three treatments in between (i.e., *k* = 16, 32, and 64) with each other. Generally, a larger number of nearest neighbors can be preferred by indoor environment, while the data partitioning and sampling configuration (i.e., block size and number of points per block) can also impact the optimal value of *k*.

Table 3.7. Performance metrics across different k values (S3DIS dataset).

Measurements	***k* = 8**	***k* = 16**	***k* = 32**	***k* = 64**	***k* = 128**
OA	84.11%	84.39%	84.39%	84.36%	84.56%
mIoU	52.16%	52.73%	52.81%	52.72%	53.04%
Ceiling	90.62%	90.69%	90.69%	90.64%	90.75%
Floor	97.26%	97.40%	97.26%	97.30%	97.27%
Wall	68.25%	68.82%	68.91%	68.80%	69.29%
Beam	46.60%	46.71%	47.07%	47.24%	47.53%
Column	7.28%	7.83%	7.37%	8.20%	8.32%
Window	38.53%	38.67%	39.08%	38.68%	39.43%
Door	60.75%	62.18%	62.65%	61.54%	63.62%
Table	60.81%	61.06%	61.13%	61.48%	61.61%
Chair	64.27%	64.96%	64.28%	64.41%	64.87%
Sofa	7.47%	8.99%	9.21%	9.85%	9.39%
Bookcase	44.17%	44.97%	45.15%	44.89%	45.48%
Board	42.75%	43.65%	44.00%	42.45%	42.01%
Clutter	49.28%	49.62%	49.69%	49.90%	49.93%

**k*: number of nearest neighbors. OA: Overall Accuracy. mIoU: mean Intersection over Union. The class name (e.g., Sofa) indicates the IoU for each class. The values are averaged across the repetitions.

Table 3.7 indicates that configuring *k* can impact the performance of the 3D deep learning model. In indoor environments, where objects can be closely spaced, the model benefits from considering a larger number of neighbors for its spatial autocorrelation features (also supported by Figure 3.11).

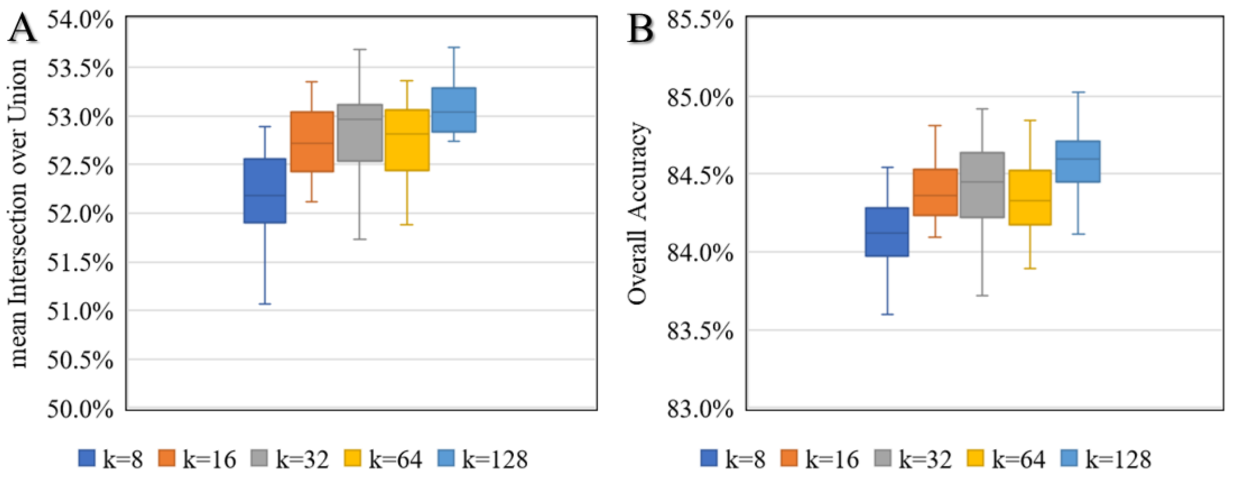


Figure 3.11. Box plot of performance measures for S3DIS dataset with Matheron semivariance. A: mean Intersection over Union; B: Overall Accuracy. Centerline is for median.

3.6.3.3 Insights for Number of Nearest Neighbor in Different Scenarios

Our findings on global performance in different environment settings indicate that optimal value of *k* values depend on the relative scale of the dataset. Scale defines the resolution and extent (Goodchild 2011). When it comes to our case, the relative scale is with respect to the two parameters configured during data preprocessing (see details in Section 3.4 Data), block size (i.e., extent) and number of points per block (i.e., related to resolution). The relative scale of the block needs to be carefully configured to fit different datasets since the objects have various spatial extents and details across datasets. For example, an indoor dataset commonly demands a smaller-scale block (1 meter in this case). For an outdoor dataset, a larger-scale block is commonly used, for instance, 8 meters in the study. The number of points per block is 4,096, which were set the same for the two datasets in this study for comparison purpose. Therefore, the actual 3D spatial extent of the 8 nearest neighbors in the outdoor dataset should be larger than that of 8 neighbors in the indoor dataset, given the same configuration on the number of points per block. In such as setting, large-scale outdoor environments tend to benefit from smaller *k* values, whereas indoor environments generally perform better with larger *k* values. Our findings can be explained as follows. In outdoor datasets, such as Semantic3D, the relative scale of a point cloud subset is typically larger, capturing a broader range of contextual information. For these datasets, a smaller *k* often effectively captures local context. In contrast, for small-scale indoor settings like S3DIS, the more limited spatial extent may require a larger *k* to better represent the local context. In summary, we demonstrated how the model performance reacts to the number of nearest neighbors, along with the datasets in different relative scales. Although the optimal value of *k* may vary among datasets, these insights offer valuable guidance for future research in configuring appropriate value of *k* for their datasets.

In the rest of this section, we investigate how varying *k* affects the performance of models on specific classes within the two different datasets. The aim is to understand whether different classes and different environment settings (large-scale outdoor versus small-scale indoor) have distinct optimal values of *k* for achieving the best model performance. While the investigation is not to suggest good value of *k* for all datasets, it is intended to provide insights into the how the performances on different classes response to the impact from *k* nearest neighbor and the relative scale of the block. Even though the current method may not be able to customize a *k* value for a specific class, the experiment results and discussion may help inform future methodological design for better performance.

The IoU of each class was averaged over 10 repetitions and aggregated at various *k* values. Figure 3.12 and Figure 3.13 show these metrics, indicating the number of points per class and illustrating the IoU differences in comparison to baselines.

For Semantic3D datasets, the four dominant classes (i.e., buildings, man-made terrain, high vegetation, and natural terrain) appear to prefer less *k* nearest neighbors (i.e., k = 8, see Figure 3.12). Car seems to be outstandingly detected when k = 64, where *k* from 8 to 32 seems to have a similar IoU value, indicating that each class prefers a specific *k*. It appears k = 128 is too big in this large-scale outdoor scenario, where it seems not to be preferred by any of the classes. We observed that there can be an upper limit of *k* that is informative in a particular environment setting, where semivariance may not be functioning over a specific limit. An extreme scenario is that the influence of semivariance will disappear if *k* equals the number of points in this block (i.e., 4,096 in this study) since all points have the same values.

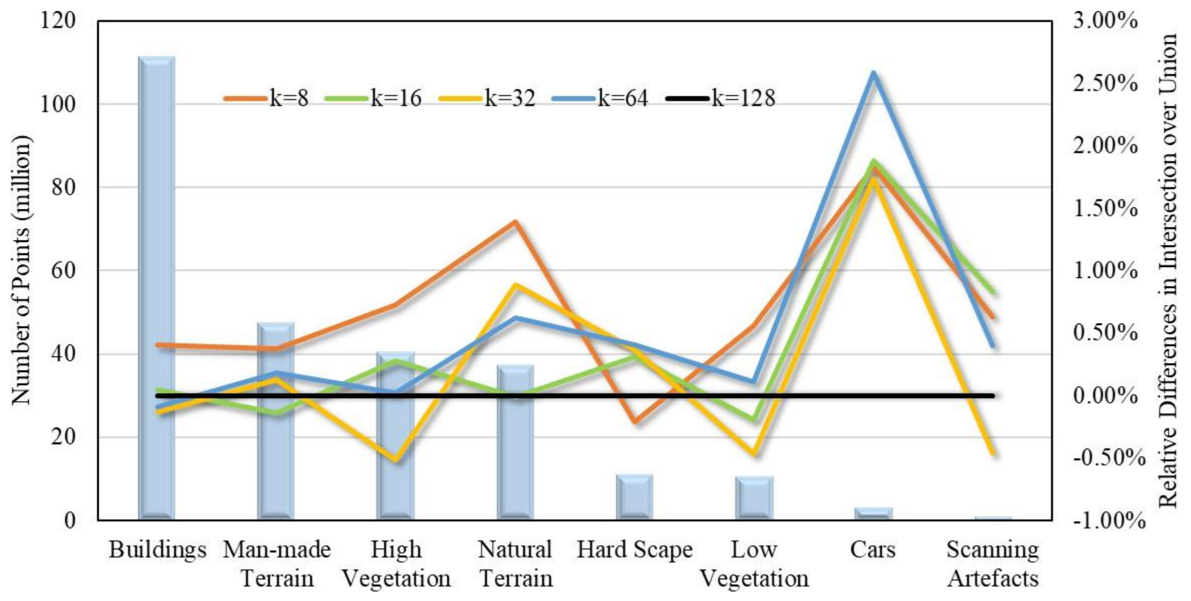


Figure 3.12. Relative differences in Intersection over Union (colored lines) along the increase of k nearest neighbors (k =128 as baseline) for Semantic3D dataset. Number of points are shown in columns. The values are averaged across the repetitions.

For S3DIS dataset, the trend of *k* against IoU (Figure 3.13), seems to be opposite from that of Semantic3D benchmark (see Figure 3.12) because a larger *k* is preferred. The IoU of the two dominant classes (i.e., ceiling and floor) seem to be insensitive to a change in *k*, which, to our understanding, could be potentially attributed to either the sufficient training data for them or the less variation of textural information. Others in this dataset seem to prefer *k* = 128 most of the time except for the board and sofa. We assume there can be a lower limit of *k* that the corresponding semivariance will be less informative if *k* is smaller than a specific threshold. For example, the semivariance within eight nearest neighbors in a small block can be close to zero.

Above all, the impact of the number of neighboring points on prediction performance may vary for different classes and datasets. Certain classes derive greater benefits from the local context—such as car in the Semantic3D dataset and door in the S3DIS dataset—whereas others do not exhibit the same level of sensitivity. The approach outlined in our study lacks the capability to dynamically adjust the number of neighbors for each class individually. However, the presented findings serve as an insightful indicator for guiding the development of future methodologies. These insights underscore the importance of considering class-specific characteristics in the design of more adaptable and efficient neighbor-selection algorithms.

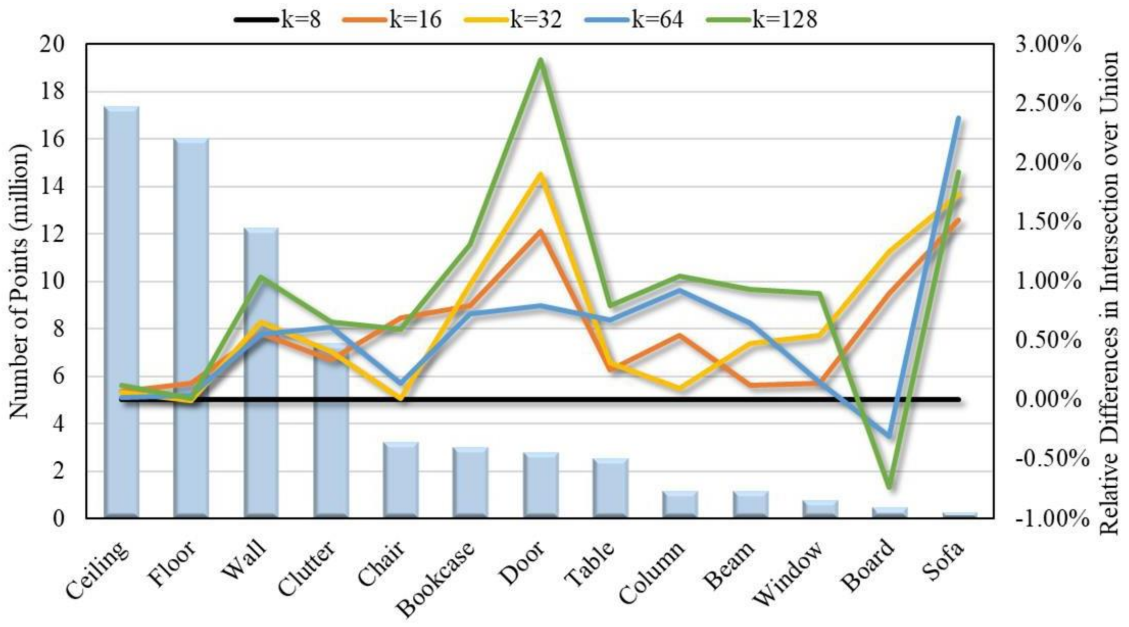


Figure 3.13. Relative differences in Intersection over Union (colored lines) along the increase of k nearest neighbors (k = 8 as baseline) for S3DIS dataset. Number of points are shown in columns. The values are averaged across the repetitions.

3.6.3.4 Discussion on Estimators for Spatial Autocorrelation Features

In our experiments, Matheron semivariance and Dowd semivariance show quite similar results, where Matheron results seem to be slightly higher (around 1%) than that of the Dowd results (see Tables A11 and A12). Moreover, they also show similar patterns with an increasing *k* (ee Dowd's in Figures A2 and A3), where there is a decreasing performance when incorporating more neighbors. We attribute this similarity to the relatively low color variability in the lag distance in the experiments as the Dowd semivariance estimator was originally proposed to derive a more robust semivariance by using the median instead of mean which is sensitive to extreme outliers (Dowd 1984). The results of the Dowd semivariance are presented in Appendix 3.4.

3.7 Conclusion

This study sheds new light on the potential of spatial autocorrelation features, specifically semivariance serving as local spatial context, in enhancing 3D deep learning's ability to understand complex 3D point cloud data with additional channels (i.e., color information in this study). Our observations have emphasized the importance of spatial autocorrelation features in refining 3D deep learning models. Its integration has resulted in noticeable improvements in per class accuracy across different environment settings.

Our first finding demonstrates the significance of spatial autocorrelation features in 3D deep learning for geospatial object detection. Specifically, the improved performance observed when incorporating color information and spatial autocorrelation features, as compared to using only spatial information, reinforces that the explicit incorporation of spatial autocorrelation enhances the power of 3D deep learning models for geospatial object detection. The spatial inner continuity and external discontinuity broadly exist in remotely sensed geospatial data no matter whether it is 2D or 3D. Moreover, the results from the examination of 3D point cloud data in this study and previous studies on 2D geospatial data suggest its ability to inform machine learning models to distinguish different objects.

In this study, we recognized the constraints of conventional methodologies for deriving semivariance from geospatial data, especially when applied to spatially unstructured and uneven distributed 3D point cloud. We have introduced a novel framework that seamlessly integrates kNN searching with the quantile binning method. This framework adeptly addressed the challenges presented by the unstructured nature of 3D point cloud, ensuring effective extraction of spatial autocorrelation features.

Our exploration into model generalization further validated the insights we presented as our third main contribution. We have provided insightful guidance for upcoming research by understanding the uncertainty of spatial context. This understanding is crucial for deriving spatial autocorrelation features. A key aspect of this process involves recognizing the sensitivities tied to the number of nearest neighbors. These sensitivities vary notably depending on the environmental settings. For instance, there are distinct considerations for indoor versus outdoor scenes. In a future study, we will keep seeking the potential of spatial autocorrelation features in helping identify 3D geospatial objects, especially finding a way to adaptively select the number of nearest neighbors for better performance.

This study not only underpins the potential of spatial autocorrelation features such as semivariance in transforming the way 3D deep learning interprets complex 3D geospatial data but also underscores the efficacy of our proposed framework. It guides future studies to further enhance the power of spatial autocorrelation in 3D deep learning—i.e., the use of spatial information or a spatial algorithm to inform deep learning algorithms while using the latter to resolve spatial problems. Embracing this bridge between deep learning and GIScience is not just an advancement for GeoAI; it represents a significant advancement in a broader field of geospatial applications.

Reference

Armeni, I., S. Sax, A. R. Zamir & S. Savarese. 2016. Joint 2D-3D-semantic data for indoor scene understanding. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1534-1543. Las Vegas, NV, USA.

Atkinson, P. M. & P. Lewis (2000) Geostatistical classification for remote sensing: An introduction. *Computers & Geosciences,* 26**,** 361-371.

Bian, L. & J. Lee (2005) Incorporating multiple index vectors and feedback to improve urban objects retrieval. *GIScience & Remote Sensing,* 42**,** 97-112.

Bian, L. & Z. Xie (2004) A spatial dependence approach to retrieving industrial complexes from digital images. *The Professional Geographer,* 56**,** 381-393.

Boulch, A. (2020) ConvPoint: Continuous convolutions for point cloud processing. *Computers & Geosciences,* 88**,** 24-34.

Cormen, T. H., C. E. Leiserson, R. L. Rivest & C. Stein. 2022. *Introduction to algorithms*. MIT press.

Couclelis, H. (1986) Artificial intelligence in geography: Conjectures on the shape of things to come. *The Professional Geographer,* 38**,** 1-11.

Cressie, N. & D. M. Hawkins (1980) Robust estimation of the variogram: I. *Journal of the International Association for Mathematical Geology,* 12**,** 115-125.

Dowd, P. A. 1984. The variogram and Kriging: Robust and resistant estimators. In *Geostatistics for Natural Resources Characterization: Part 1,* eds. G. Verly, M. David, A. G. Journel & A. Marechal, 91-106. Dordrecht: Springer Netherlands.

Duan, W., Y.-Y. Chiang, S. Leyk, J. H. Uhl & C. A. Knoblock (2020) Automatic alignment of contemporary vector data and georeferenced historical maps using reinforcement learning. *International Journal of Geographical Information Science,* 34**,** 824-849.

Durrieu, S. & R. F. Nelson (2013) Earth observation from space – the issue of environmental sustainability. *Space Policy,* 29**,** 238-250.

Engelmann, F., T. Kontogianni, A. Hermans & B. Leibe. 2017. Exploring spatial context for 3D semantic segmentation of point clouds. In *Proceedings of the IEEE international conference on computer vision workshops*, 716-724.

Fan, S., Q. Dong, F. Zhu, Y. Lv, P. Ye & F.-Y. Wang. 2021. SCF-net: Learning spatial contextual features for large-scale point cloud segmentation. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 14504-14513. IEEE.

Goodchild, M. (2022) The Openshaw effect. *International Journal of Geographical Information Science,* 36**,** 1697-1698.

Goodchild, M. & W. Li (2021) Replication across space and time must be weak in the social and environmental sciences. *Proceedings of the National Academy of Sciences,* 118.

Goodchild, M. F. (2011) Scale in GIS: An overview. *Geomorphology,* 130**,** 5-9.

Guo, Y., H. Wang, Q. Hu, H. Liu, L. Liu & M. Bennamoun (2021) Deep learning for 3D point clouds: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 43**,** 4338-4364.

Haack, B. N., N. D. Herold, M. A. Bechdol & Others (2000) Radar and optical data integration for land-use/land-cover mapping. *Photogrammetric Engineering and Remote Sensing,* 66**,** 709-716.

Hackel, T., N. Savinov, L. Ladicky, J. D. Wegner, K. Schindler & M. Pollefeys (2017) Semantic3D.net: A new Large-scale Point Cloud Classification Benchmark. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences,* IV-1-W1.

Haralick, R. M., K. Shanmugam & I. h. Dinstein (1973) Textural features for image classification. *IEEE Transactions on Systems, Man, and Cybernetics,* SMC-3**,** 610-621.

Humeau-Heurtier, A. (2019) Texture feature extraction methods: A Survey. *IEEE Access,* 7**,** 8975-9000.

Jaderberg, M., K. Simonyan & A. Zisserman (2015) Spatial transformer networks. *Conference on Neural Information Processing Systems,* 28.

Kamal, M., S. Phinn & K. Johansen (2014) Characterizing the spatial structure of mangrove features for optimizing image-based mangrove mapping. *Remote Sensing,* 6**,** 984-1006.

Karasiak, N., J. F. Dejoux, C. Monteil & D. Sheeren (2022) Spatial dependence between training and test sets: another pitfall of classification accuracy assessment in remote sensing. *Machine Learning,* 111**,** 2715-2740.

Kattenborn, T., J. Leitloff, F. Schiefer & S. Hinz (2021) Review on Convolutional Neural Networks (CNN) in vegetation remote sensing. *ISPRS Journal of Photogrammetry and Remote Sensing,* 173**,** 24-49.

Klemmer, K., N. S. Safir & D. B. Neill. 2023. Positional encoder graph neural networks for geographic data. In *International Conference on Artificial Intelligence and Statistics*, 1379-1389. PMLR.

Kwan, M.-P. (2012) The uncertain geographic context problem. *Annals of the Association of American Geographers,* 102**,** 958-968.

Landrieu, L. & M. Simonovsky. 2018. Large-scale point cloud semantic segmentation with superpoint graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 4558-4567.

Lee, D. R. & G. T. Sallee (1970) A method of measuring shape. *Geographical Review***,** 555-563.

Li, Z., M. E. Hodgson & W. Li (2018) A general-purpose framework for parallel processing of large-scale LiDAR data. *International Journal of Digital Earth,* 11**,** 26-47.

Matheron, G. (1963) Principles of geostatistics. *Economic geology,* 58**,** 1246-1266.

Maturana, D. & S. 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)*, 922-928. ieeexplore.ieee.org.

Miranda, F. P. & J. R. Carr (1994) Application of the semivariogram textural classifier (STC) for vegetation discrimination using SIR‐B data of the guiana shield, northwestern brazil. *Remote Sensing Reviews,* 10**,** 155-168.

Miranda, F. P., L. E. N. Fonseca & J. R. Carr (1998) Semivariogram textural classification of JERS-1 (Fuyo-1) SAR data obtained over a flooded area of the Amazon rainforest. *International Journal of Remote Sensing,* 19**,** 549-556.

Miranda, F. P., J. A. Macdonald & J. R. Carr (1992) Application of the semivariogram textural classifier (STC) for vegetation discrimination using SIR-B data of Borneo. *International Journal of Remote Sensing,* 13**,** 2349-2354.

Openshaw, S. (1992) Some suggestions concerning the development of artificial intelligence tools for spatial modelling and analysis in GIS. *The annals of regional science,* 26**,** 35-51.

Openshaw, S. & C. Openshaw. 1997. *Artificial intelligence in geography*. John Wiley & Sons, Inc.

Pereira, E., E. Silveira, I. T. Bueno & F. W. A. Júnior (2019) Spatial and spectral remote sensing features to detect deforestation in Brazilian Savannas. *Advances in Forestry Science,* 6**,** 775-782.

Qi, C., H. Su, K. Mo & L. J. Guibas. 2017a. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition,* 652-660.

Qi, C., H. Su, M. Nießner, A. Dai, M. Yan & L. J. Guibas. 2016. Volumetric and multi-view cnns for object classification on 3D data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 5648-5656.

Qi, C., L. Yi, H. Su & L. J. Guibas (2017b) Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in Neural Information Processing Systems,* 30.

Rezatofighi, H., N. Tsoi, J. Gwak, A. Sadeghian, I. Reid & S. Savarese. 2019. Generalized intersection over union: A metric and a loss for bounding box regression. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition,* 658-666.

Shekhar, S., P. R. Schrater, R. R. Vatsavai, W. Wu & S. Chawla (2002) Spatial contextual classification and prediction models for mining geospatial data. *IEEE Transactions on Multimedia,* 4**,** 174-188.

Simonovsky, M. & N. Komodakis. 2017. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition,* 3693-3702.

Smith, T. R. (1984) Artificial intelligence and its applicability to geographical problem solving. *The Professional Geographer,* 36**,** 147-158.

Tang, W. & M. Jia (2014) Global sensitivity analysis of a large agent-based model of spatial opinion exchange: A heterogeneous multi-GPU acceleration approach. *Annals of the Association of American Geographers,* 104**,** 485-509.

Tang, W. & S. Wang (2020) Navigating high performance computing for geospatial applications. *High Performance Computing for Geospatial Applications***,** 1-5.

Tso, B. & R. C. Olsen (2004) Algorithms and technologies for multispectral, hyperspectral, and ultraspectral imagery x. *Algorithms and Technologies for*.

Wu, W., Z. Qi & L. Fuxin. 2019. Pointconv: Deep convolutional networks on 3d point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 9621-9630.

Wu, X., J. Peng, J. Shan & W. Cui (2015) Evaluation of semivariogram features for object-based image classification. *Geo-spatial Information Science,* 18**,** 159-170.

Yu, M. & Q. Liu (2021) Deep learning-based downscaling of tropospheric nitrogen dioxide using ground-level and satellite observations. *Science of the Total Environment,* 773**,** 145145.

Zawadzki, J., C. Cieszewski, M. Zasada & R. Lowe (2005) Applying geostatistics for investigations of forest ecosystems using remote sensing imagery. *Silva Fennica,* 39.

Zhao, H., L. Jiang, J. Jia, P. H. S. Torr & V. Koltun. 2021. Point Transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision,* 16259-16268. openaccess.thecvf.com.

Appendix 3.1 Modified Implementation of PointNet

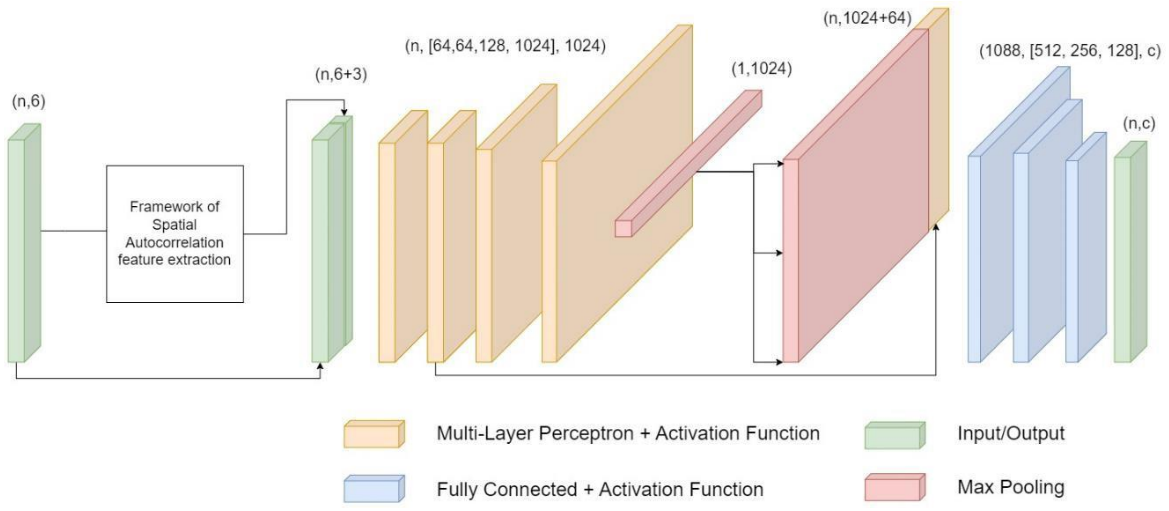

(n,6)
Framework of Spatial Autocorrelation feature extraction
(n,6+3)
(n, [64,64,128, 1024], 1024)
(1,1024)
(n,1024+64)
(1088, [512, 256, 128], c)
(n,c)
Multi-Layer Perceptron + Activation Function
Input/Output
Fully Connected + Activation Function
Max Pooling

Figure A1. Modified PointNet architecture incorporating spatial autocorrelation features as additional channels for large-scale semantic segmentation. The numbers on top of the layers suggest the number of input channels, number of neurons per layer, and number of outputs channel. n is the number of points per block and c is the number of classes.

Appendix 3.2 Summary of the Statistics for Accuracy Measurements in Experiment 1

Table A1. Results of 10 repetitions for models trained on spatial information only on Semantic3D dataset.

Statistics	**Mean**	**Std.**	**Max**	**Min**
OA	66.49%	2.97%	69.26%	59.59%
mIoU	30.74%	1.80%	33.35%	28.48%
Man-made terrain	69.18%	6.23%	77.29%	58.40%
Natural terrain	27.03%	8.50%	38.27%	14.60%
High vegetation	36.80%	3.52%	44.09%	31.15%
Low vegetation	8.58%	2.68%	11.60%	2.65%
Buildings	65.64%	5.31%	70.65%	55.02%
Hard scape	11.80%	3.79%	18.65%	6.47%
Scanning artefacts	12.60%	1.66%	16.45%	10.74%
Cars	14.29%	4.77%	20.05%	6.43%

*OA: Overall Accuracy. mIoU: mean Intersection over Union.

Table A2. Results of 10 repetitions for models trained on spatial information and color information on Semantic3D dataset.

Statistics	**Mean**	**Std.**	**Max**	**Min**
OA	81.95%	1.06%	83.10%	79.66%
mIoU	51.58%	1.56%	53.30%	47.84%
Man-made terrain	90.69%	0.77%	91.75%	89.02%
Natural terrain	72.59%	4.15%	77.64%	64.96%
High vegetation	57.01%	2.33%	61.81%	53.54%
Low vegetation	24.12%	1.38%	25.64%	20.75%
Buildings	79.85%	1.63%	81.60%	77.22%
Hard scape	21.55%	3.08%	26.44%	14.68%
Scanning artefacts	20.25%	2.69%	23.89%	14.23%
Cars	46.55%	4.49%	51.44%	36.98%

*OA: Overall Accuracy. mIoU: mean Intersection over Union

Table A3. Results of 10 repetitions for models trained on datasets with additional spatial autocorrelation information on Semantic3D dataset.

Statistics	**Mean**	**Std.**	**Max**	**Min**
OA	83.32%	1.17%	84.86%	81.38%
mIoU	54.05%	1.83%	56.52%	51.55%
Man-made terrain	91.02%	0.85%	92.23%	89.58%
Natural terrain	73.45%	2.37%	78.30%	68.82%
High vegetation	60.15%	3.32%	63.54%	54.51%
Low vegetation	26.41%	2.09%	30.25%	23.36%
Buildings	81.38%	1.88%	83.91%	78.24%
Hard scape	23.19%	1.52%	25.15%	20.33%
Scanning artefacts	25.10%	2.90%	28.52%	20.38%
Cars	51.72%	5.28%	59.07%	42.40%

*OA: Overall Accuracy. mIoU: mean Intersection over Union

Table A4. One-tailed t-test results in terms of p-value for treatments on Semantic3D dataset.

Statistics	Spatial info. v.s. RGB	RGB v.s. Semivariance
OA	<0.01	0.01
mIoU	<0.01	<0.01
Man-made terrain	<0.01	0.19
Natural terrain	<0.01	0.29
High vegetation	<0.01	0.01
Low vegetation	<0.01	<0.01
Buildings	<0.01	0.03
Hard scape	<0.01	0.07
Scanning artefacts	<0.01	<0.01
Cars	<0.01	0.01

*****Spatial info., RGB, and Semivariance are the three treatments aforementioned.

Table A5. Results of 10 repetitions for models trained on spatial information only on S3DIS dataset.

Statistics	**Mean**	**Std.**	**Max**	**Min**
OA	79.80%	0.27%	80.07%	79.21%
mIoU	45.31%	0.34%	45.74%	44.64%
Ceiling	87.02%	0.69%	88.08%	85.94%
Floor	96.68%	0.32%	96.93%	96.16%
Wall	59.92%	0.52%	61.01%	59.32%
Beam	44.80%	1.32%	46.72%	42.65%
Column	9.82%	1.61%	13.58%	7.73%
Window	19.45%	2.78%	23.40%	13.76%
Door	37.86%	0.83%	39.71%	36.51%
Table	56.88%	0.84%	58.39%	55.48%
Chair	62.99%	2.01%	65.47%	59.08%
Sofa	26.05%	2.24%	28.47%	22.11%
Bookcase	35.13%	1.80%	38.16%	33.17%
Board	9.99%	0.82%	11.19%	8.46%
Clutter	42.40%	0.53%	43.18%	41.43%

*OA: Overall Accuracy. mIoU: mean Intersection over Union.

Table A6. Results of 10 repetitions for models trained on spatial information and color information on S3DIS dataset.

Statistics	**Mean**	**Std.**	**Max**	**Min**
OA	82.83%	0.51%	83.47%	81.90%
mIoU	50.19%	0.72%	51.05%	48.99%
Ceiling	89.03%	0.45%	89.80%	88.36%
Floor	97.14%	0.08%	97.27%	97.04%
Wall	65.99%	1.33%	67.89%	63.43%
Beam	45.50%	0.99%	47.17%	43.95%
Column	5.99%	1.09%	7.61%	4.83%
Window	37.40%	1.34%	39.15%	35.29%
Door	58.78%	1.93%	62.19%	56.18%
Table	59.25%	1.67%	61.43%	56.38%
Chair	61.03%	1.89%	63.32%	56.38%
Sofa	6.33%	1.49%	9.28%	4.93%
Bookcase	40.20%	2.07%	43.63%	36.27%
Board	38.54%	3.14%	42.38%	32.40%
Clutter	47.35%	1.27%	48.65%	44.68%

*OA: Overall Accuracy. mIoU: mean Intersection over Union.

Table A7. Results of 10 repetitions for models trained on datasets with additional spatial autocorrelation information on S3DIS dataset.

Statistics	**Mean**	**Std.**	**Max**	**Min**
OA	84.19%	0.33%	84.78%	83.55%
mIoU	52.54%	0.56%	53.38%	51.34%
Ceiling	90.82%	0.60%	91.53%	89.63%
Floor	97.18%	0.21%	97.41%	96.75%
Wall	68.54%	0.70%	69.87%	67.31%
Beam	46.90%	1.18%	48.85%	44.86%
Column	7.72%	1.36%	9.51%	5.33%
Window	39.04%	1.04%	40.10%	37.09%
Door	61.64%	2.56%	67.14%	58.98%
Table	60.51%	1.11%	62.89%	58.71%
Chair	64.29%	0.71%	65.29%	62.84%
Sofa	9.00%	4.57%	17.91%	3.42%
Bookcase	44.29%	1.36%	46.29%	41.89%
Board	43.98%	1.54%	46.21%	41.82%
Clutter	49.07%	0.96%	50.46%	47.44%

*OA: Overall Accuracy. mIoU: mean Intersection over Union.

Table A8. One-tailed t-test results for treatments on Semantic3D dataset in terms of p-value.

Statistics	Spatial info. v.s. RGB	RGB v.s. Semivariance
OA	<0.01	<0.01
mIoU	<0.01	<0.01
Ceiling	<0.01	<0.01
Floor	<0.01	0.26
Wall	<0.01	<0.01
Beam	0.13	0.01
Column	<0.01	<0.01
Window	<0.01	0.01
Door	<0.01	<0.01
Table	<0.01	0.04
Chair	0.03	<0.01
Sofa	<0.01	0.05
Bookcase	<0.01	<0.01
Board	<0.01	<0.01
Clutter	<0.01	0.01

*****Spatial info., RGB, and Semivariance are the three treatments aforementioned.

Appendix 3.3 One-tailed t-test Results for Different kNN

Table A9. One-tailed t-test results for Semantic3D dataset in terms of *p*-value.

kNN	OA	mIOU
8-16	0.14	0.17
8-32	0.05	0.07
8-64	0.29	0.34
8-128	0.11	0.06
128-8	0.11	0.06
128-16	0.30	0.18
128-32	0.49	0.35
128-64	0.21	0.09

*kNN: k nearest neighbor. OA: Overall Accuracy. mIoU: mean Intersection over Union.

Table A10. One-tailed t-test results on S3DIS dataset in terms of *p*-value.

kNN	**OA**	**mIOU**
8-16	<0.01	<0.01
8-32	<0.01	<0.01
8-64	0.01	<0.01
8-128	<0.01	<0.01
128-8	<0.01	<0.01
128-16	0.02	0.02
128-32	0.05	0.05
128-64	<0.01	0.01

*kNN: k nearest neighbor. OA: Overall Accuracy. mIoU: mean Intersection over Union.

Appendix 3.4 Results of Perturbing *k* Nearest Neighbors Using Dowd Estimator

Table A11. Performance metrics across different *k* values (Semantic3D dataset) using Dowd estimator.

Measurements	***k* = 8**	***k* = 16**	***k* = 32**	***k* = 64**	***k* = 128**
OA	82.99%	82.94%	83.06%	82.91%	82.66%
mIoU	53.36%	53.12%	53.22%	52.99%	52.71%
Man-made terrain	90.90%	90.72%	90.93%	90.59%	90.78%
Natural terrain	72.95%	73.52%	73.78%	72.01%	72.51%
High vegetation	60.47%	60.19%	60.33%	60.53%	59.39%
Low vegetation	25.56%	25.69%	25.23%	25.21%	25.74%
Buildings	81.31%	81.35%	81.51%	81.47%	81.06%
Hard scape	22.62%	22.52%	22.50%	22.00%	21.89%
Scanning artefacts	24.43%	22.87%	23.55%	23.42%	22.78%
Cars	48.66%	48.11%	47.96%	48.72%	47.52%

**k*: number of nearest neighbors. OA: Overall Accuracy. mIoU: mean Intersection over Union. The class name (e.g., Cars) indicates the IOU for each class. The values are averaged across the repetitions.

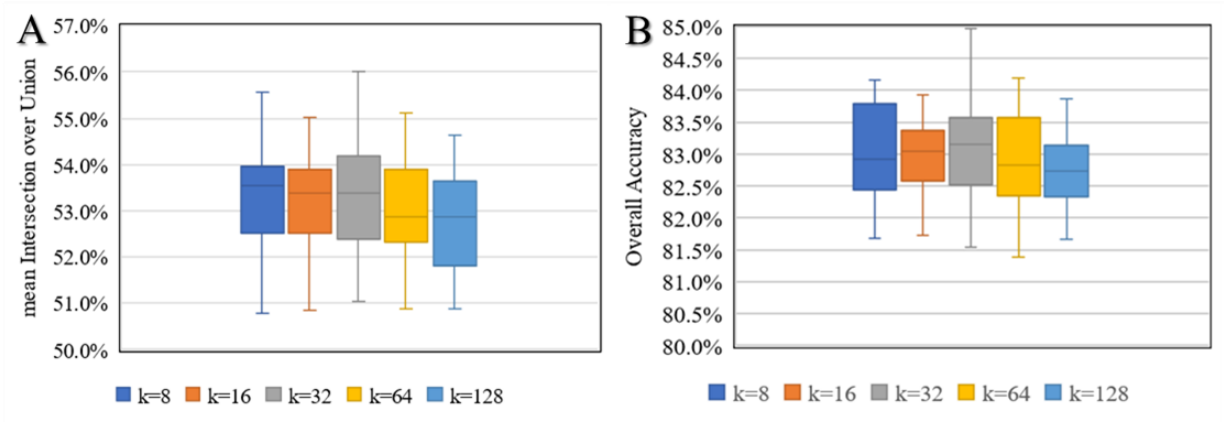


Figure A2. Box chart of performance measures for Semantic3D dataset with Dowd estimator. A: mean Intersection over Union; B: Overall accuracy. Centerline is for median.

Table A12. Performance metrics across different *k* values (S3DIS dataset) using Dowd estimator.

Measurements	***k* = 8**	***k* = 16**	***k* = 32**	***k* = 64**	***k* = 128**
OA	83.86%	84.14%	84.16%	84.29%	84.37%
mIoU	51.74%	52.30%	52.48%	52.56%	52.82%
Ceiling	90.40%	90.54%	90.44%	90.62%	90.57%
Floor	97.26%	97.24%	97.27%	97.32%	97.28%
Wall	67.74%	68.39%	68.31%	68.79%	68.77%
Beam	46.11%	46.69%	47.27%	47.26%	47.22%
Column	6.48%	7.18%	7.56%	8.02%	7.99%
Window	37.63%	37.80%	38.37%	38.59%	38.70%
Door	60.12%	61.71%	61.49%	61.85%	62.70%
Table	60.43%	60.76%	60.39%	60.48%	61.10%
Chair	63.68%	64.19%	63.87%	64.31%	64.69%
Sofa	8.41%	9.47%	10.02%	9.40%	10.31%
Bookcase	44.17%	44.97%	45.15%	44.89%	45.48%
Board	42.75%	43.65%	44.00%	42.45%	42.01%
Clutter	49.28%	49.62%	49.69%	49.90%	49.93%

**k*: number of nearest neighbors. OA: Overall Accuracy. mIoU: mean Intersection over Union. The class name (e.g., Sofa) indicates the IOU for each class. The values are averaged across the repetitions.

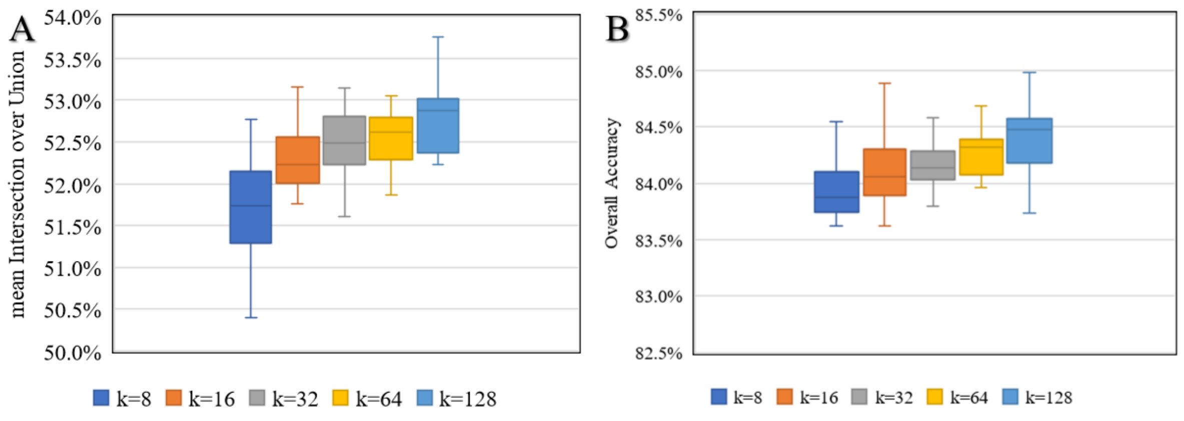


Figure A3. Box plot of performance measures for S3DIS dataset with Dowd estimator. A: mean Intersection over Union; B: Overall accuracy. Centerline is for median.

4 SPATIAL AUTOCORRELATION ENCODER FOR 3D DEEP LEARNING

4.1 Introduction

3D geospatial object detection is underscored in its critical role in building accurate 3D models for state-of-the-art applications of geographical information science (GIScience). These applications span digital earth (Guo, Goodchild and Annoni 2020), twin cities (Goodchild 2022b, Batty 2023), and Building Information Modeling (BIM) (Goodchild 2021, Batty 2013), where the 3D representations are important not only for visualizing but also for analyzing, and management. In the domain of GIScience, the past decades have witnessed a remarkable evolution in 3D techniques, ranging from the development of data acquisition technologies (e.g., LiDAR[8]) to the evolution of data processing and analyzing technologies supported by computing technologies. Early studies naively represent 3D spatial objects, essentially reflecting the spatial location of the object in a 3D space. For example, objects were represented by spatial points in a 3D network (Kwan and Lee 2005) to describe the spatial relationship among them. Moreover, the representations of buildings are simply derived from blueprints (Evans, Hudson-Smith and Batty 2006, Batty and Hudson - Smith 2005, Batty 2000) without the as-built status of them. However, nowadays, the advancements of 3D techniques have made it possible to generate up-to-date, and as-is representations of diverse geospatial objects, laying the foundation for the development of 3D geographical information systems (GIS). Therefore, there is a demand for accuracy and efficiency for 3D geospatial object detection.

An illustrative example of this is Tree Folio NYC[9], which is a digital twin of New York City (NYC) Urban Canopy produced by the Design Across Scales Lab at Cornell University. It is a web-based

[8] LiDAR stands for light detection and ranging.
[9] https://labs.aap.cornell.edu/daslab/projects/treefolio

GIS application designed to provide practitioners and stakeholders with a user-friendly platform for querying, analyzing, and visualizing the 3D point cloud representations of individual trees in NYC. The development of such applications requires the detection and extraction of trees from 3D point clouds. Due to the number of trees in NYC (approximately 8 million), it would become incredibly time-consuming and labor-intensive work if performed manually.

3D deep learning algorithms can be potentially used to address this challenge. Deep learning for object detection in 3D context has been attracted unprecedented focus since the first architecture, PointNet (Qi et al. 2017a), which is a deep neural network designed to directly consume point cloud as input, shaping the development of neural network architectures in recent years. The variants of this architecture have been continuously serving as a key part in many cutting-edge architectures from recent studies (Xie et al. 2021, Wu et al. 2020, Ren et al. 2024, Qian et al. 2022). Parallel to these advancements, the emergence of GeoAI—a synthesis of GIScience and Artificial Intelligence (AI)—marks a pivotal shift towards not only using cutting-edge AI methodologies to inform geographical studies but also to enrich AI with geographical insights (Goodchild 2022a, Li 2021). A few efforts have been seen since then. For example, Chen (2024) conducted a systematic investigation and proved the effectiveness of semivariance as a representation of spatial autocorrelation in informing 3D deep learning.

Semivariance is essentially estimated by a function, such as Matheron's estimator (Matheron 1963) and Dowd’s estimator (Dowd 1984), on pairwise differences in a neighborhood. Even though there are enhancement brought by explicitly feeding the semivariance into the model for object detection, there are two weaknesses of the method that may entangle the practitioners to use it. One is that pre-calculation of semivariance is required, which demands additional effort towards object detection rather than end-to-end[10]. The other weakness is that it is difficult for users especially those without expert knowledge to well configure the parameters, such as number of nearest neighbors, number of bins, and semivariance estimators, whose optimal setting might differ for various datasets.

Therefore, this study proposed a spatial autocorrelation encoder, which is a neural network-based module to extract high-dimensional vector as a representation of spatial contextual features for the neighborhood of each point based on pairwise differences ordered by spatial lag distance. The proposed spatial autocorrelation encoder neither requires the user to pre-calculate the contextual features for each point, nor demands expert knowledge to carefully configure the dataset-dependent parameters for the model. The proposed encoder directly extracts a context embedding from the ordered pairwise difference and the parameters are configured during the training process. This study also conducted environments to investigate the effectiveness of ordered pairwise differences to prove the validity in using it as the initial representation of 3D spatial autocorrelation.

The proposed encoder not only enhances the capability of GIS in handling and interpreting 3D geospatial data but also paves the way for further investigation on geographical insights in benefiting AI models. The implications of this are significant for various applications, including urban planning, environmental monitoring, and disaster management, where quick and accurate interpretation of spatial data is crucial. The contributions of this study are highlighted as follows:

- Enhanced Geospatial Object Detection: The study underscores the utility of ordered pairwise differences for identifying diverse geospatial objects, reinforcing the synergy

[10] End-to-end, in the domain machine learning, typically refers to a process or a model that takes raw data as input and directly produces the expected output, without demanding any manual intermediate steps operated by humans.

between geographic theories, statistical methods, and deep learning advancements. It demonstrates the pivotal role of spatial statistics in enriching AI technologies for geospatial object detection from complex environments.

- Automate Contextual Representation Extraction: By developing a neural network-based encoder that effectively extracts spatially contextual embeddings from ordered pairwise differences, this research showcases an innovative integration of AI in geospatial analysis. This approach simplifies the application of traditional semivariance estimations, offering a streamlined, dataset-specific learning mechanism that enhances model accuracy and efficiency in geospatial object detection.

The remainder of this manuscript is organized as follows: Section 4.2 reviews the related literature with a focus on contextual features in object detection. Sections 4.3 to 4.5 detailly explained the methodology of this study, including the design of the proposed encoder, processing to derive ordered pairwise difference, dataset used in this study, and experimental designs. Section 4.6 delineated the results of the experiments followed by discussion about them. Finally, the conclusions are depicted in Section 4.7.

4.2 Literature Review

Spatial context is important to the task of 3D object detection (Mottaghi et al. 2014, Pohlen et al. 2017, Engelmann et al. 2017). To improve the performance of deep learning architectures on 3D datasets, many studies focus on using different feature extraction modules to improve the deep learning model. We listed some of them from a scope of geographical insights enhancing AI model. Wu et al. (2020) presents an advanced method for 3D object detection in point clouds, improving upon the Frustum PointNet (Charles et al. 2018) by incorporating local neighborhood information into point feature computation. This approach enhances the representation of each point through the neighboring features. The novel local correlation-aware embedding operation leads to superior detection performance on the KITTI dataset compared to the F-PointNet baseline. This method emphasized the importance of local spatial relationships for 3D object detection in deep learning frameworks. Klemmer, Safir and Neill (2023) adds a positional encoder using Moran's I as an auxiliary task to enhance the graphic neural network in interpolation tasks. Fan et al. (2021) designed a module based on the distance between points to capture its local spatial context to inform object detection. Engelmann et al. (2017) proposed a network to incorporate larger-scale spatial context in order to improve the model performance by considering the interrelationship among subdivisions (i.e., blocks) of the point cloud. While most of the endeavors contribute to improving the model capability to derive discriminative feasters from spatial information, information from other channels seems to be overlooked (Chen 2024). Current LiDAR often captures more information rather than the position, such as intensity, and RGB, while some special sensor can further collect other spectrum information such as near inferred for thermal studies. Even though some studies integrate RGB as input (Qi et al. 2017b, Qi et al. 2017a, Boulch 2020), colors are not their focus and limited considerations are taken on utilizing them. Chen (2024) pioneered a study with a focus on making more use of RGB information by explicitly incorporating semivariance variables to improve the model performance, which are estimated based on the variation of observed color information and corresponding spatial lag distance. Extending the scope of Chen (2024), we would further explore the spatial variances of non-spatial information. The color information as well as the texture embedded in it, for example, is important for humans to recognize different objects (Haralick, Shanmugam and Dinstein 1973, Tso and Olsen 2004). Therefore, we emphasize the importance of color in geospatial object detection, moreover, we call for investigation of the channels other than spatial information and corresponding endeavor to improve the performance of models. We are leading a focus shift towards bridging the gap between the development of 3D deep learning and better utilizing non-spatial information.

4.3 Methodology

4.3.1 Architecture design of the spatial autocorrelation encoder

Inspired by Chen (2024) explicitly feeding semivariance as contextual features for object detection, we aim to automate the estimation process and derive dataset-dependent contextual embeddings by using a spatial autocorrelation encoder (see Figure 4.1). One important requirement for the design of the neural network architecture for 3D deep learning is that the embeddings should be permutation invariant to the input (Qi et al. 2017). PointNet uses max pooling as a symmetric function that makes the output global signatures invariant in terms of permutation. ConvPoint (Boulch 2020) adopted a continuous convolution operation to ensure the convoluted features are permutation invariant to the input point cloud. The spatial dependency embeddings extracted by the proposed encoder are naturally permutation invariant to the input point cloud because the input unordered point cloud for the contextual feature estimation will be resorted based on the lag distance (see Figure 4.2).

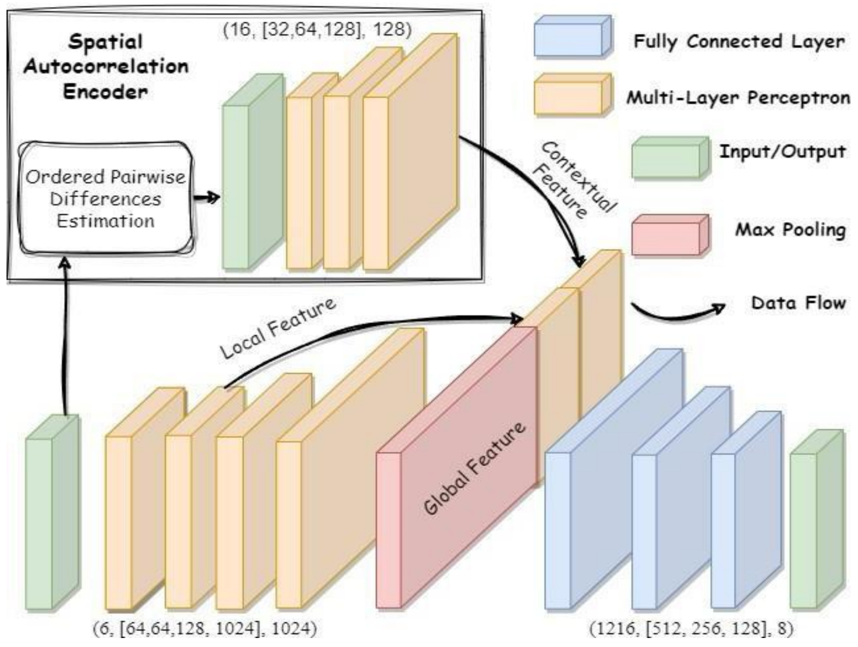


Figure 4.1. Proposed spatial autocorrelation encoder for spatial autocorrelation features extraction. PointNet is used as the basic structure for demonstration purposes. The numbers in parenthesis are the # input channels, # neurons in hidden layers, and # output channels.

4.3.2 Ordered pairwise differences.

Ordered pairwise differences is the initial input of the sub-neural network from the spatial autocorrelation encoder. We demonstrate how to derive ordered pairwise difference using one point as example shown in Figure 4.2. To calculate ordered pairwise differences, we would need to define the neighbors. Due to the unstructured nature of a 3D point cloud, it is not feasible to use a fixed distance to identify the neighbors. For example, the number of neighbors within a certain distance can vary across different points. Moreover, one point in point sparse areas may not find a neighbor within a given distance. Therefore, it is essential to use k nearest neighbor to identify the neighborhood. This idea is supported by many cutting-edge studies, such as PointNet++ (Qi et al. 2017b), PointNext (Qian et al. 2022), and ConvPoint (Boulch 2020) using the kNN approach. kNN is also adopted in this paper. One argument that is often associated along with kNN is that the local neighborhood captured by kNN varied its size for different points, which may lead to inconsistent features. However, we are aware of this difficulty brought by kNN that feature might not be inconsistent, but we address this challenge through random sampling as suggested by ConvPoint. Randomly sampling points to represent the same object within the same scene in the training process is a way to mitigate the inconsistencies brought by the kNN approach.

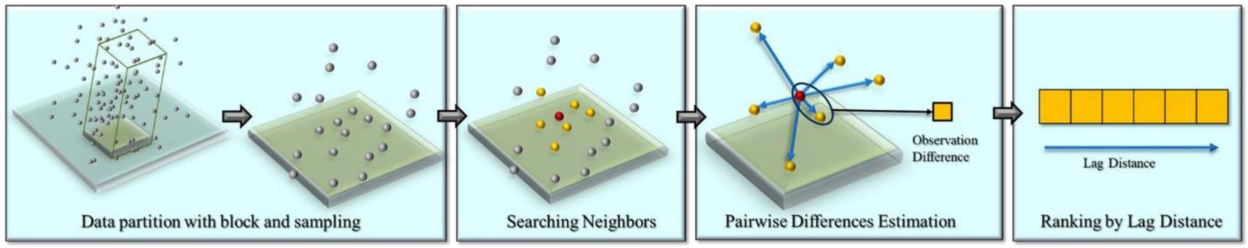


Figure 4.2. Illustration of the preprocessing to prepare a training dataset and derive ordered pairwise differences. One point from a block is used for demonstration purpose.

To make it more clear how the original point clouds are prepared, we illustrate how the pairwise differences are calculated from the initial dataset. Data partitioning and subsampling are two essential steps in the preprocessing steps of cutting-edge 3D deep learning methods. This is due to the consideration related to computing resources and the nature of neural networks. The point cloud datasets directly collected by sensors like LiDAR instruments commonly include millions of points with the dataset often gigabytes in size, which is too large to be efficiently, or feasibility be fed to the CPU and/or GPU. On the other hand, neural networks require structured input, but the unstructured nature of 3D point clouds does not satisfy this requirement. Therefore, we have to perform data partitioning and subsampling to prepare structured and memory-manageable datasets for deep neural networks. These two steps are demonstrated as the first two steps in Figure 4.2. Data partitioning is implemented by using a particular size of blocks to subtract a subset from the original point cloud. Furthermore, a sampling method is conducted to sample from the subset to an anticipated number. Considering the unevenly distributed nature of 3D point cloud, this step is often done by sampling with replacement. This is to overcome the case that the number of points within the subset does not reach the expected number of points per block.

A block of sampled points is the input for the ordered pairwise differences extraction. We used one point from such a block as an example to explain how the pairwise differences are extracted. First, we identify the neighbors of the point by kNN. While kNN is computationally intensive, especially in current 3D deep learning methods that employ **k** , it takes around 1 ⁄ 3 of the total computing time to train a deep learning model. We used tree searching embedded in SciPy[11] to implement kNN, which is more efficient than calculating all pair distances. Once k nearest neighbors are identified, the pairwise differences are calculated between the center point (key point) and its neighbors. In the traditional calculation of semivariance, the pairwise differences not only consider the key point and its neighbors within total n values. The total number of that is n*(n-1) differences because all pair differences are calculated among the identified neighbors. We chose to use the former design because it is simpler in computational complexity, and it is as powerful as the latter one in identifying different objects in our preliminary experiments. Finally, we ordered the k nearest neighbors by its lag distance away from the key points, using this as a representation of initial context features of the point. Subsequently, we applied this to all points in the block to prepare a dataset. We chose 16 as the k value to identify the nearest neighbor in this study, referring to the empirical setting from the previous studies when aggregating features from neighbors (Fan et al. 2021, Boulch 2020).

[11] https://scipy.org/

4.3.3 Feature grouping

In a deep neural network, each feature extraction layer can extract the features from the previous layer (either input or hidden layers). In such a way, high-level features (from later layers) can be eventually extracted, while the final abstract features should be adequately invariant to most local changes from early layers (e.g., input layer). In our design, we are trying to extract such high-level embedding features from pairwise differences within a neighborhood and use these features along with embedding features from spatial and color information for the final prediction. The k nearest neighbors will be found for each point and pairwise differences will be calculated based on the values of the neighbors as demonstrated in Figure 4.2. The ordered pairwise differences will be fed to the spatial autocorrelation encoder (see Figure 4.1). Finally, the high-level embedding features will be concatenated for classification.

Aggregating local features to a larger scale (up to global scale) can effectively improve the model performance. By incorporating the spatial autocorrelation module, we would group the spatial dependency embeddings to other features of a neural network. There are three types (Qi et al. 2017b) of grouping templates (three settings of framework), single-scale point grouping (SSG), multi-scale point grouping (MSG), and multi-resolution point grouping (MRG). Single-scale point grouping (SSG) is used to extract abstraction by layers and only use the final layer features as the feature of a point for classification. Multi-resolution point grouping (MRG) is extremely computationally expensive (Qi et al. 2017b), in which larger scale features are derived based on features from smaller scale. Multi-scale point grouping (MSG) appears to be a simple but effective approach to group layers from different scales. The final features of points combine features from different scales, where features for each scale can be independently derived. The grouping templates can also help to mitigate the impact from uneven distribution of point clouds. Multi-scale and single scale grouping, claimed by (Engelmann et al. 2017), appear to have similar performance, where MSG is 2.1% higher than SS in terms of IoU. Therefore, we followed an MSG template to group the features. We concatenated the local contextual features with the global and point-wise local features (see Figure 4.1).

4.4 Dataset

Semantic3D, introduced by Hackel et al. (2017), stands as a substantial and diverse dataset specifically tailored for outdoor scene analysis. We selected this dataset because its context is close to GIS applications such as twin city, where it can serve as the original data for 3D modeling geospatial objects, such as buildings, trees, traffic lights, and road surface. The Semantic3D dataset offers a detailed and complex dataset with 15 scenes ranging from urban to rural. It covers a wide range of eight semantic categories, man-made and natural terrains, high and low vegetation, structures like buildings and hardscape (e.g., road light, and fencing), scanning artefacts (e.g., dynamic noise during scanning), and vehicles (see Figure 4.3).

As we are moving towards the concept of twin cities, where urban environments are enriched with sensors and technology for better management and planning, the need for accurate and efficient processing of 3D spatial data becomes increasingly critical (Batty 2023, Guo et al. 2020). The ability to accurately segment and interpret this data can inform various aspects of smart city planning, including infrastructure development, environmental monitoring, and emergency response strategies (Kwan and Lee 2005, Batty 2023, Li, Batty and Goodchild 2020, Batty 2013, Batty 2008, Evans et al. 2006). The Semantic3D benchmark serves as a bridge between academic research and real-world applications. It provides a common ground for researchers to test and compare their methodologies, fostering an environment of collaboration and continuous improvement. This is particularly important in fast-evolving fields like GIS, where the gap between theoretical research and practical application needs to be constantly narrowed.

In our methodology, we carefully chose nine of these scenes for our training dataset, selected in terms of their diversity. The other six scenes were used for validation to examine the generalization capability of our model. For preprocessing, we adopted an 8-meter block size for dataset partitioning, aligning with recommendations of Boulch (2020), and targeted a density of 4,096 points per block. A block size of 8 meters indicates that the dataset is divided by an 8-meter grid, where each block covers an area of 8 meters by 8 meters. This setting is a good rule of thumb for large scale outdoor dataset, which is also supported by (Tang et al. 2022). This segmentation facilitates the handling and analysis of large datasets by breaking them down into more manageable units, allowing for detailed processing and analysis of each segment while maintaining the structural nature of spatial data. To maintain the quality of ordered pairwise difference estimation, blocks containing fewer than 128 unique points were excluded. This approach addresses the issue of duplicated points in oversampled blocks by prioritizing unique point locations, ensuring that our model input is not skewed by artificial data replication. For example, when estimating the ordered pairwise differences, only unique points are taken into consideration, otherwise, oversampled points can count towards k nearest neighbors, which can lead a weak representation of spatial relationships within the dataset for accurate geospatial object detection.

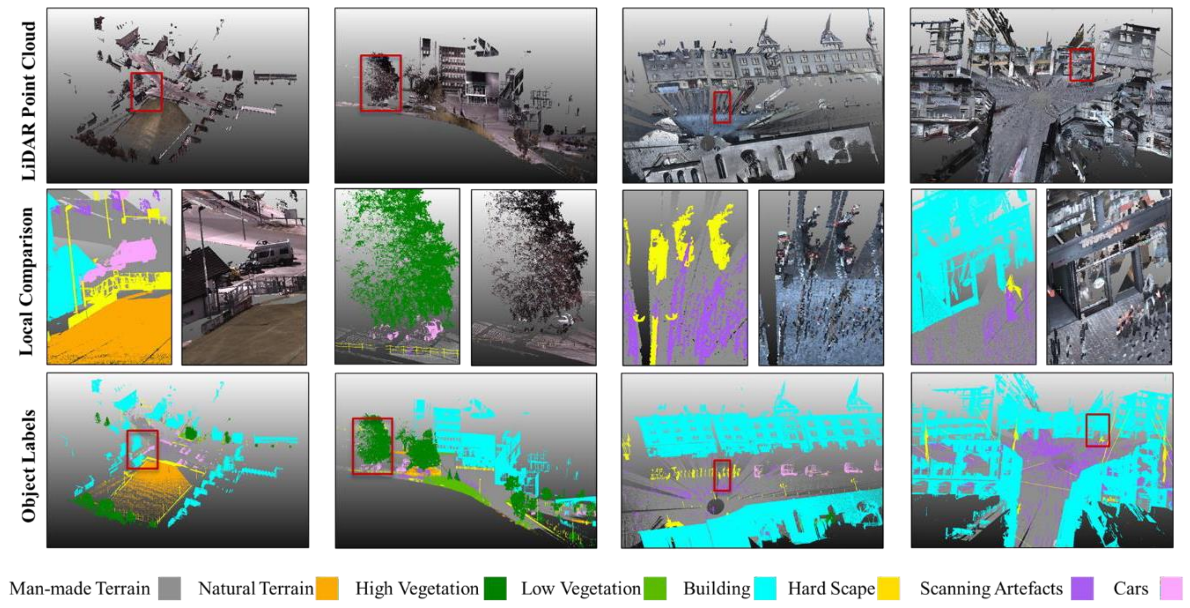


Figure 4.3. Demonstration of Semantic3D benchmark dataset.

There are two main challenges in the dataset, uneven point distribution and a long tail problem. The points within these scenes are not uniformly distributed; instead, they exhibit an extremely uneven spatial distribution as is the nature of LiDAR data. This unevenness poses a unique challenge as it requires algorithms to be highly adaptable and sensitive to a wide variety of spatial contexts and densities. Furthermore, the classes exhibit extremely uneven distribution, known as a long tail problem. As suggested in Figure 4.4, the first four largest classes (i.e., building, man-made terrain, high vegetation and natural terrain) represent approximately 90% of the whole dataset. In particular, the building class is approximately 45% of the points in the entire dataset. The remaining four classes, hard scape, low vegetation, cars, and scanning artefacts only represent <10% proportion of the dataset. The ability to adequately represent non-uniform point-cloud data are essential for developing sophisticated 3D deep learning models that can accurately interpret and interact with complex and variable real-world environments. The Semantic3D dataset, therefore, serves as an invaluable resource for advancing research and development in 3D scene analysis and understanding.

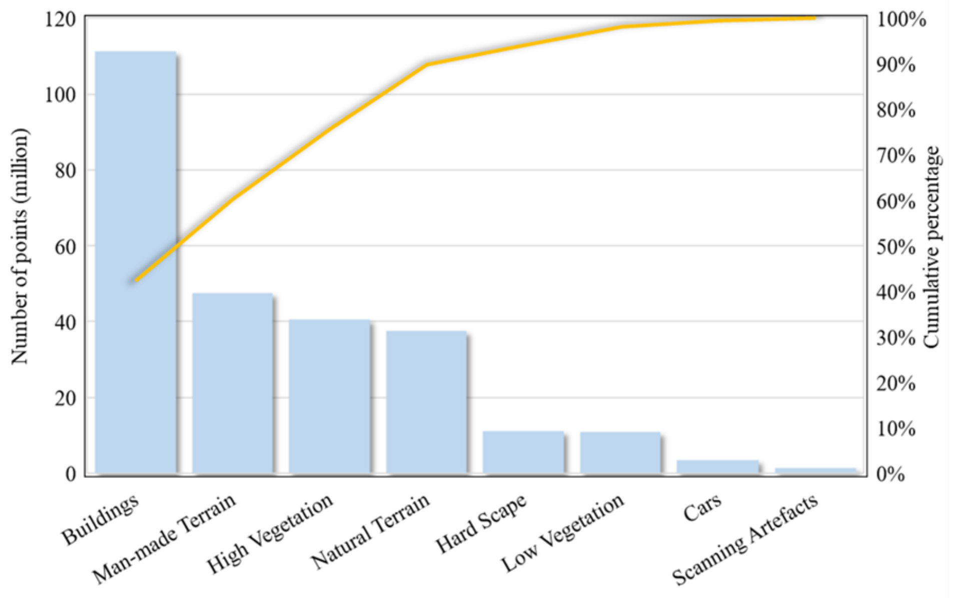


Figure 4.4 Distribution of different classes in Semantic3D dataset.

4.5 Experiment

We designed two experiments, where one is to investigate the effectiveness of ordered pairwise difference for object detection and the other is to examine the effectiveness of the proposed architecture to inform the 3D deep learning for geospatial object detection.

The first experiment is to investigate the usefulness of ordered pairwise difference in identifying different geospatial objects using 3D deep learning methods. Practically, we only feed pairwise differences to a 3D deep neural network except any spatial information or RGB data. We used PointNet here not only because it is concise and powerful to derive good representations (Guo et al. 2021) but also because of its compatibility to handle any type of input features. Cutting edge architectures have a more sophisticated design requiring an explicit feed of spatial information.

For example, neural network architectures designed by Boulch (2020) and Fan et al. (2021) have an essential need of explicit spatial information as an input. In this case, we would not be able to solely examine the effectiveness of ordered pairwise differences for distinguishing geospatial objects.

The second experiment aims to assess the effectiveness of the designed architecture In benefiting the 3D deep learning in terms of the object detection capability. Moreover, we will compare the results with models directly trained on a combination of spatial information, color information, and contextual information.

To address the uncertainties introduced by the stochastic process during the training, we trained and validated the model for each treatment with 10 repetitions. The number of repetitions was set by considering the computational challenges against our available computing resources. The validation dataset was pre-generated so that the performance measurements were comparable. Then, we summarized the averaged performance measurements over the 10 repetitions, where Intersection over Union (IoU), mean Intersection over Union (mIoU), Average Accuracy (AA) and Overall Accuracy (OA) were used as indicators to identify how well the model generalizes on a validation dataset. The average performance metrics across the 10 repetitions were calculated for each treatment. Subsequently, comparisons were conducted to assess the observed differences. One tailed t-test is applied to examine if their mean is significantly different. The equations of the three metrics are shown below. By employing the systematic approach, this experiment aims to offer an in-depth understanding of the effectiveness of the module to inform 3D deep learning.

$$IoU = \frac{TP_i}{TP_i + FN_i + FP_i} \tag{4.1}$$

$$AA = mean(\frac{TP_i}{TP_i + FN_i}) \tag{4.2}$$

$$mIoU = mean(IoU_i) \tag{4.3}$$

$$OA = \frac{TP_{all}}{TP_{all} + FP_{all}} \tag{4.4}$$

where *TP*, *FP*, and *FN* are True Positive, False Positive, and False Negative; subscript *all* means across all classes and *i* refers to each individual class.

4.6 Results and Discussion

4.6.1 Computing performance of the experiments

In this study, to address the computational challenges brought by either hyperparameter tuning or repeated models training, we utilized high-performance computing (HPC) cluster resources equipped with Graphics Processing Units (GPUs). HPC has been widely applied in geospatial research to tackle computational challenges, as noted by Tang and Wang (2020), and the integration of GPUs, as mentioned by Tang and Jia (2014), offers remarkable capabilities for solving geospatial issues. Our HPC cluster comprises 10 nodes, each equipped with an NVIDIA A40 GPU, an Intel Xeon Gold 6326 CPU with 8 cores, and 64 GB of memory. This setup significantly saved hundreds of hours for hyper-parameter tuning in this study so that we can have an adequate configuration for Experiment 2. The computing time of Experiments 2 shrunk from 126.8 hours as sequential time to 13.4 hours, when utilizing all 10 nodes in parallel. This represents a speed-up of approximately 9.46 times. Consequently, the GPU-enhanced HPC resources provide robust support for the computational demands of our 3D deep learning model training in the experiments conducted in this study.

4.6.2 Investigating effectiveness of ordered pairwise difference.

Experiment 1 investigated the effectiveness of ordered pairwise differences in terms of geospatial object detection. The measures are reported in Table 4.1 and demonstration of the results are shown in Figure 4.5. It is interesting to see the overall accuracy can achieve 63% while the input data is only the ordered pairwise difference without any explicit spatial locations or RGB information, which supports a moderate capability of pairwise difference in identifying geospatial objects. Even though it does not show a good performance in all classes, it still has an adequate performance for several of them, such as building and high vegetation. This performance is especially visible on selected scenes as shown in Figure 4.5.

Table 4.1. Performance results for Experiment 1.

OA	63%
AA	40%
mIOU	28%
Man-made terrain	39%
Natural terrain	38%
High vegetation	54%
Low vegetation	8%
Buildings	63%
Hard scape	12%
Scanning artefacts	0%
Cars	11%

*OA: Overall Accuracy. AA: Average Accuracy. mIoU: mean Intersection over Union. The class name (e.g., Cars) indicates the IoU for each class. The values are averaged across the repetitions.

The IoU for each class shows a big difference across classes. The model performance on buildings and high vegetation separately reaches 63% and 54% in IoU. The second tier of classes are man-made terrain and natural terrain, on which the model has an IoU of 39% and 38% correspondingly. The rest of classes, hard scape, low vegetation, cars, and scanning artefacts, are not well detected by the model, where the IoU ranges from 0% to 12%.

We observed that the model performance on different classes seems to be related to the proportion of the class within the dataset and the volume of the objects. As shown in Figure 4.3, buildings, man-made terrain, natural terrain, and high vegetation are the four largest classes, where a cumulative proportion is around 90% of the whole dataset. The model shows better performance on these four classes and appears far better than the rest of classes in terms of IoU. One potential reason that causes the diversity performance of the model on each class is the proportion of the class within the dataset. Even though man-made terrain has more points than high vegetation class, the performance on high vegetation is better than that of man-made terrain. It seems ordered pairwise differences have the capability to identify different objects while it also appears to be sensitive to the volume of the objects. We attribute this finding to the nature of data partitioning and sampling process. While in this process, the number of points for objects in the original dataset directly impacts how many points from this object can be captured during this process. Moreover, the volume of the object also impacts the analysis. For example, a cubic object (e.g., building and tree) tends to be captured with more points than the planar one since ground only exists on the floor surface of a block but cubic objects exist across the space. Hardscape, scanning artefacts, and cars take a small volume as well as the number of points as opposed to the whole scene. Therefore, they might be less represented using the pairwise differences as the points sampled for them might be more subject to boundary effects.

Even though there are less represented classes, we still innovatively find the capability of ordered pairwise difference as context features to identify many objects. We can tell from Figure 4.5 that buildings, trees and terrains, even though confusion between natural terrain and man-made terrain, are adequately identified by the model. The cutting-edge results from a model based on spatial information and RGB surpasses the performance of the ordered pairwise difference (Boulch 2020, Fan et al. 2021), but knowing the usefulness of pairwise difference is also an important finding.

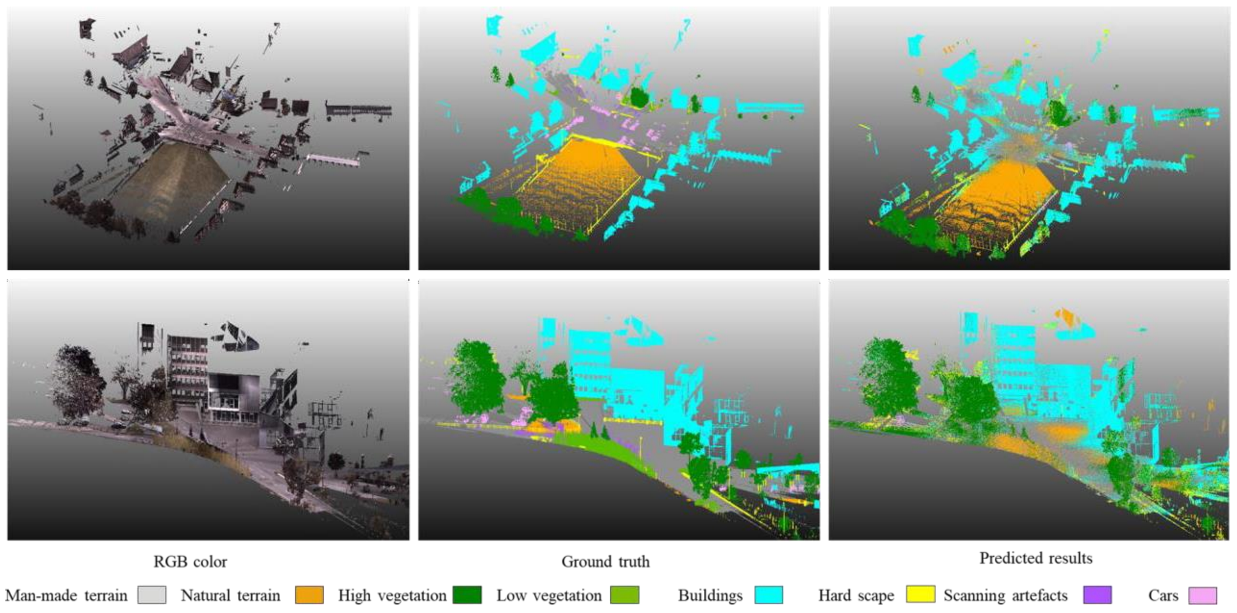


Figure 4.5. Visualization of the prediction results of using order pairwise differences only as the input. Ordered pairwise differences only show an adequate performance on objects with a large volume in the scene, such as high vegetation, natural/man-made terrain, and buildings.

The visualization of the prediction results demonstrated the confusion among classes especially around the boundaries. For example, the natural terrain and man-made terrain are not well differentiated by the model. Moreover, we also observed that some man-made surfaces are predicted as trees if the points are close to a tree. These results represent a challenge of current method to that spatial autocorrelation features of those points on boundaries may not adequately be represented since the neighboring points can be from other classes. This problem is also seen in the classification task on 2D remotely sensed imagery when spatial autocorrelation is considered. Myint (2003) tried to mitigate this challenge by excluding the samples on the boundaries during model training. Wu et al. (2015) addressed the boundary issue by considering the object-based spatial contextual features instead of the window-based one. However, further studies have to be conducted to adapt these solutions from 2D to 3D content so that this challenge can be well addressed.

4.6.3 Performance of spatial autocorrelation encoder

Experiment 2 is to investigate the effectiveness of the designed spatial autocorrelation encoder. Once we derived the ordered pairwise differences based on the partitioned block of point with a fixed number of points, they were fed separately to a sub neural network for feature encoding. The features were grouped with global features and point features from a normal 3D deep neural network, PointNet in our case. In this section we demonstrated the measurement statistics across the 10 repetitions (see detail in Appendix 4.1) using the measures delineated in Table 4.2. We further demonstrate our results in Figure 4.6 comparing with the results without the encoder.

The OA, AA, and mIOU are 85.5%, 69.7%, and 57.6% separately, providing a global measure for the model performance for this dataset. The IoU across the classes ranges from 26.8% (scanning artefacts) to 92.8% (man-made terrain). Same as reflected in Experiment 1, the classes with higher proportion from the original datasets tend to have a better performance. For example, man-made terrain (IoU 92.8%), buildings (IoU 84.0%), natural terrain (IoU 78.7%), and high vegetation (IoU 66.1%) are the four classes that the model best performed on, while they are also the four largest classes with a cumulative proportion of 90% of the original dataset. We attribute their better performance to the amount of data compared to the rest of the classes. The model shows a moderate performance on cars with 55.6% IoU. Scanning artefacts, hard scape, and low vegetation seems to be not well detected by the model with IoU ranging from 26.8% to 28.8%.

Table 4.2. Statistical results across 10 repetitions.

Statistics	**Mean**	**Std.**	**Max**	**Min**
OA	85.5%	0.2%	85.1%	85.8%
AA	68.7%	0.5%	68.1%	69.6%

mIoU	57.6%	0.4%	56.9%	58.4%
Man-made terrain	92.8%	0.3%	92.4%	93.1%
Natural terrain	78.7%	1.4%	76.4%	80.4%
High vegetation	66.1%	1.2%	63.9%	68.3%
Low vegetation	28.8%	2.6%	25.7%	33.2%
Buildings	84.0%	0.4%	83.0%	84.7%
Hard scape	27.7%	0.9%	26.1%	28.9%
Scanning artefacts	26.8%	1.6%	23.9%	29.0%
Cars	55.6%	1.8%	52.6%	57.8%

*OA: Overall Accuracy. AA: average accuracy mIoU: mean Intersection over Union. The class name (e.g., Cars) indicates the IOU for each class. The values are averaged across the repetitions.

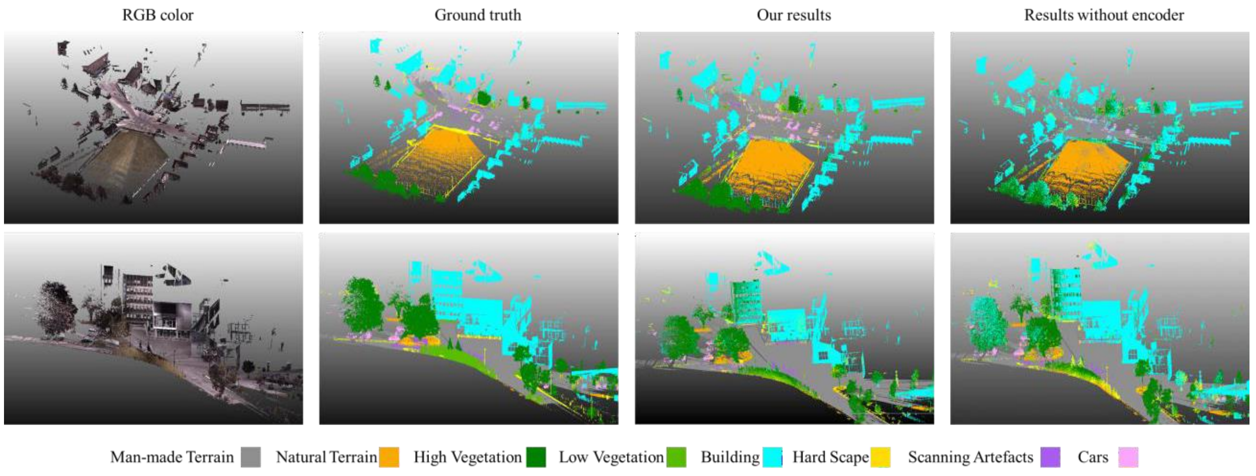


Figure 4.6. Comparison between ground truth, and the predicted results w./w.o. spatial autocorrelation encoder.

4.6.4 Comparison analysis

We conducted a comparison analysis between the results from our study with the one in Chen (2024). Chen (2024) systematically analyzed how the input feature will impact the model performance on geospatial object detection, where the three treatments are spatial information only, spatial information and RGB, and one with additional semivariances. While, it has been shown that the latter two produced a significant increase in accuracy as compared to the use of spatial information on its own in the analysis. In this comparison analysis, we compare our results with results from Chen (2024) for treatments including RGB and additionally combining semivariances. We use RGB as a baseline to investigate how context information can inform the model for geospatial object detection and a one tailed t-test is performed against the baseline to explore its statistical significance (see Table 4.3).

Table 4.3. Comparison analysis results.

Statistics	RGB	Semivariance (gain, *p*-value)	Our study (gain, *p*-value)
OA	81.95%	83.32% (+1.37%, 0.01)	85.55% (+3.60%, < 0.01)
AA	64.03%	66.04% (+2.01%, 0.01)	68.72% (+4.69%, < 0.01)
mIOU	51.58%	54.05% (+2.47%, 0.00)	57.58% (+6.00%, < 0.01)
Man-made terrain	90.69%	91.02% (+0.33%, 0.19)	92.78% (+2.09%, < 0.01)
Natural terrain	72.59%	73.45% (+0.86%, 0.29)	78.67% (+6.08%, < 0.01)
High vegetation	57.01%	60.15% (+3.14%, 0.01)	66.10% (+9.09%, < 0.01)
Low vegetation	24.12%	26.41% (+2.29%, < 0.01)	28.81% (+4.69%, < 0.01)
Buildings	79.85%	81.38% (+1.53%, 0.03)	84.05% (+4.20%, < 0.01)
Hard scape	21.55%	23.19% (+1.64%, 0.07)	27.73% (+6.18%, < 0.01)
Scanning artefacts	20.25%	25.10% (+4.85%, < 0.01)	26.83% (+6.58%, < 0.01)
Cars	46.55%	51.72% (+5.17%, 0.01)	55.64% (+9.09%, < 0.01)

*OA: Overall Accuracy. AA: average accuracy mIoU: mean Intersection over Union. The class name (e.g., Cars) indicates the IOU for each class. The values are averaged across the repetitions. RGB and semivariance represent the two treatments from Chen (2024).

For the global measurements, OA, AA, and mIOU, there is an increasing trend in accuracy as we consider spatial context with more sophisticated measures. Without considering the spatial content, the three measurements are 81.95%, 64.03%, and 51.58% using RGB data on its own. Explicitly incorporating semivariance as a spatial context feature can result in a 1-3% increase in the global measurements. Once we applied a neural network to directly learn a spatial context embedding from ordered pairwise differences, accuracy improved 3.60%, 4.69%, and 6.00% for the three assessment measures with the increase being significant at 99% confidence interval.

The comparison analysis significantly underscores the advancement our study brings to the field of geospatial object detection using 3D deep learning. By directly learning spatial context embeddings from ordered pairwise differences, our approach outperforms previous models trained on RGB and/or semivariance in terms of OA, AA, and mIOU. The substantial gains across various classes, particularly in categories like high vegetation and cars, highlight the effectiveness of incorporating spatial autocorrelation features. The comparison results validate the advancement of the proposed encoder over previous approaches in deriving spatial contextual features in geospatial object detection tasks. The statistical significance of these results emphasizes the critical role of spatial contextual features in enhancing 3D deep learning models, paving the way for future advancements in this rapidly evolving field.

4.7 Conclusion

This study introduces a spatial autocorrelation encoder to integrate spatial contextual features into 3D deep learning for enhancing object detection within point clouds. By leveraging ordered pairwise differences, the encoder significantly improves the accuracy of geospatial object detection, especially in complex urban and natural environments. Experimental results underscore the effectiveness of this approach, suggesting its potential for a wide range of applications in GIS and smart city planning. Moreover, it also supported the effectiveness of ordered pairwise differences in geospatial object detection, which can stand alone to have adequate performance on some of the objects. This finding is innovative and suggests the potential use of pairwise differences in future improvements in geospatial object detection. Furthermore, the proposed spatial autocorrelation encoder not only streamlines the workflow for geospatial object detection explicitly considering spatial autocorrelation but also simplifies the extraction of sophisticated spatial autocorrelation features, making it accessible to practitioners without deep expertise in the field.

Our study started initially from the call for integrating GIS, and AI to enhance the development of state-of-the-art GIS applications, such as digital twin projects like Tree Folio. By innovating in the realm of geospatial object detection, we provide a robust foundation for future research and applications in urban planning, environmental monitoring, and beyond. The exploration and findings presented serve not only as a remarkable progress witnessed in these domains but also as a bridge connecting the theoretical underpinnings of GIS with the practical applications of AI in 3D geospatial object detection.

Reference

Batty, M. (2000) The new urban geography of the third dimension. *Environment and Planning B: Planning and Design,* 27**,** 483-484.

--- (2008) Virtual reality in geographic information systems. *The Handbook of Geographic Information Science. Oxford, Blackwell Publishing***,** 317-334.

---. 2013. Agents, models, and geodesign. Citeseer.

--- (2023) Digital twins in city planning. *Nature Computational Science,* 4.

Batty, M. & A. Hudson ‐ Smith (2005) Urban simulacra: London. *Architectural Design,* 75**,** 42-47.

Boulch, A. (2020) ConvPoint: Continuous convolutions for point cloud processing. *Comput. Graph.,* 88**,** 24-34.

Charles, R. Q., W. Liu, C. Wu, H. Su & J. G. Leonidas. 2018. Frustum pointnets for 3D object detection from RGB-D data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 918-927.

Chen, T. C. W. T. C. A. S.-E. (2024) Spatial explicit incorporation of spatial autocorrelation features in 3D deep learning [Manuscript submitted for publication]. *Annals of the American Association of Geographers*.

Dowd, P. A. 1984. The variogram and Kriging: Robust and resistant estimators. In *Geostatistics for Natural Resources Characterization: Part 1,* eds. G. Verly, M. David, A. G. Journel & A. Marechal, 91-106. Dordrecht: Springer Netherlands.

Engelmann, F., T. Kontogianni, A. Hermans & B. Leibe. 2017. Exploring spatial context for 3D semantic segmentation of point clouds. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 716-724.

Evans, S., A. Hudson-Smith & M. Batty (2006) 3-D GIS: Virtual London and beyond. An exploration of the 3-D GIS experience involved in the creation of virtual London. *Cybergeo: European Journal of Geography*.

Fan, S., Q. Dong, F. Zhu, Y. Lv, P. Ye & F.-Y. Wang. 2021. SCF-net: Learning spatial contextual features for large-scale point cloud segmentation. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 14504-14513. IEEE.

Goodchild, M. (2022a) The Openshaw effect. *Journal of Geographical Information Science,* 36**,** 1697-1698.

Goodchild, M. F. (2021) Introduction to urban big data infrastructure. *Urban Informatics***,** 543-545.

--- (2022b) Elements of an infrastructure for big urban data. *Urban Informatics*, 1**,** 3.

Guo, H., M. F. Goodchild & A. Annoni. 2020. *Manual of digital Earth*. Springer Nature.

Guo, Y., H. Wang, Q. Hu, H. Liu, L. Liu & M. Bennamoun (2021) Deep Learning for 3D Point Clouds: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 43**,** 4338-4364.

Hackel, T., N. Savinov, L. Ladicky, J. D. Wegner, K. Schindler & M. Pollefeys. 2017. *Semantic3D.net: A new large-scale point cloud classification benchmark.*

Haralick, R. M., K. Shanmugam & I. h. Dinstein (1973) Textural Features for Image Classification. *IEEE Transactions on Systems, Man, and Cybernetics,* SMC-3**,** 610-621.

Klemmer, K., N. S. Safir & D. B. Neill. 2023. Positional encoder graph neural networks for geographic data. In *International Conference on Artificial Intelligence and Statistics*, 1379-1389. PMLR.

Kwan, M.-P. & J. Lee (2005) Emergency response after 9/11: The potential of real-time 3D GIS for quick emergency response in micro-spatial environments. *Computers, Environment and Urban Systems,* 29**,** 93-113.

Li, W. (2021) GeoAI and Deep Learning. *The International Encyclopedia of Geography*.**,** 1-6.

Li, W., M. Batty & M. F. Goodchild (2020) Real-time GIS for smart cities. *International Journal of Geographical Information Science,* 34**,** 311-324.

Matheron, G. (1963) Principles of geostatistics. *Economic Geology,* 58**,** 1246-1266.

Mottaghi, R., X. Chen, X. Liu, N.-G. Cho, S.-W. Lee, S. Fidler, R. Urtasun & A. Yuille. 2014. The role of context for object detection and semantic segmentation in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 891-898.

Myint, S. (2003) Fractal approaches in texture analysis and classification of remotely sensed data: Comparisons with spatial autocorrelation techniques and simple descriptive statistics. *International Journal of Remote Sensing,* 24**,** 1925-1947.

Pohlen, T., A. Hermans, M. Mathias & B. Leibe. 2017. Full-resolution residual networks for semantic segmentation in street scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4151-4160.

Qi, C., H. Su, K. Mo & L. J. Guibas. 2017a. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 652-660.

Qi, C., L. Yi, H. Su & L. J. Guibas (2017b) Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in Neural Information Processing Systems,* 30.

Qian, G., Y. Li, H. Peng, J. Mai, H. Hammoud, M. Elhoseiny & B. Ghanem (2022) Pointnext: Revisiting pointnet++ with improved training and scaling strategies. *Advances in Neural Information Processing Systems,* 35**,** 23192-23204.

Ren, D., Z. Ma, Y. Chen, W. Peng, X. Liu, Y. Zhang & Y. Guo (2024) Spiking PointNet: Spiking neural networks for point clouds. *Advances in Neural Information Processing Systems,* 36.

Tang, W., S.-E. Chen, J. Diemer, C. Allan, T. Chen, Z. Slocum, T. Shukla, V. S. Chavan & N. S. Shanmugam. 2022. DeepHyd: A deep learning-based artificial intelligence approach for the automated classification of hydraulic structures from LiDAR and sonar data. North Carolina Department of Transportation. Research and Development Unit.

Tso, B. & R. C. Olsen (2004) Scene classification using combined spectral, textural, and contextual information. *Algorithms and Technologies*.

Wu, C., J. Pfrommer, J. Beyerer, K. Li & B. Neubert. 2020. Object detection in 3D point clouds via local correlation-aware point embedding. In *2020 Joint 9th International Conference on Informatics, Electronics & Vision (ICIEV) and 2020 4th International Conference on Imaging, Vision & Pattern Recognition (icIVPR)*, 1-8. IEEE.

Wu, X., J. Peng, J. Shan & W. Cui (2015) Evaluation of semivariogram features for object-based image classification. *Geo-spatial Information Science,* 18**,** 159-170.

Xie, J., Y. Xu, Z. Zheng, S.-C. Zhu & Y. N. Wu. 2021. Generative pointnet: Deep energy-based learning on unordered point sets for 3d generation, reconstruction and classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 14976-14985.

Appendix 4.1 Inference Performance of 10 Repetitions

Treat	OA	AA	mIoU	Man-made terrain	Natural terrain	High vegetation	Low vegetation	Building	Hard scape	Scanning artefacts
1	85.7%	68.3%	57.7%	93.0%	79.6%	67.0%	28.4%	84.3%	28.3%	26.4%
2	85.6%	68.1%	57.3%	92.5%	78.8%	66.2%	28.3%	84.1%	28.9%	25.2%
3	85.8%	69.6%	58.4%	92.8%	80.4%	66.8%	29.7%	84.4%	28.3%	26.9%
4	85.6%	68.7%	57.5%	92.8%	78.4%	66.5%	25.8%	84.0%	27.2%	28.6%
5	85.3%	68.6%	57.4%	92.7%	76.4%	66.3%	25.7%	83.8%	27.6%	29.0%
6	85.4%	68.8%	56.9%	92.4%	78.0%	68.3%	25.7%	84.7%	26.1%	27.4%
7	85.4%	69.0%	57.6%	92.8%	77.3%	63.9%	31.2%	84.2%	28.3%	27.0%
8	85.1%	69.5%	57.1%	92.5%	77.2%	66.0%	29.4%	83.8%	26.5%	25.8%
9	85.8%	68.3%	58.0%	93.1%	80.4%	64.8%	33.2%	83.0%	28.3%	23.9%
10	85.7%	68.4%	57.9%	93.1%	80.2%	65.4%	30.8%	84.1%	28.0%	28.1%

*Treat is for treatment IDs. OA is Overall Accuracy. AA is Average Accuracy. mIoU is for mean Intersection over Union. The name of the class represents the IoU for each class.

5 SPATIAL INTERPOLATION TO ENHANCE DEEP LEARNING-BASED 3D GEOSPATIAL OBJECT DETECTION FOR LARGE-SCALE SCENES

5.1 Introduction

3D geospatial object detection has been attracting unprecedented attention, driven by an increasing availability of 3D data. This is largely attributed to innovations in 3D data acquisition technologies, including light detection and ranging (LiDAR), RGB-D sensors, and the supportive systems provided by unmanned aerial, ground, and surface vehicles. 3D geospatial object detection has seen a significant boost from recent breakthroughs in 3D deep learning techniques, which have been used for obstacle detections for self-driving vehicles, and building highly accurate 3D models for cutting-edge applications, such as digital twin cities (Batty 2023, Goodchild 2022), and Building Information Modeling (BIM) (Goodchild 2021). The demand for prediction accuracy in these applications is more focused than ever, given its potential impact on public safety.

The recent development of 3D deep learning techniques has been largely benefiting the tasks of 3D geospatial object detection. One milestone of the current 3D deep learning techniques is the study of (Qi et al. 2017a), where a deep learning architecture, PointNet, was developed to directly take raw point cloud as input. The design of the architecture, especially the symmetric function, inspired and shaped the recent development in 3D deep learning, addressing the challenge brought by the unstructured and unordered nature of point cloud data. Following this conceptual design, scholars have applied successful architectures from different tasks to 3D deep learning, where these architectures are but not limited to multi-layer perceptron (Qi et al. 2017a, Qi et al. 2017b), convolutional neural networks (Boulch 2020, Wu, Qi and Fuxin 2019), graph neural networks (Shi and Rajkumar 2020), and attention-based neural networks (Zhao et al. 2021).

In this context, the DeepHyd project (Tang et al. 2022) emerges as a pivotal development of a deep learning-based application used for a particular domain, hydrology. DeepHyd is specifically designed for the detection of hydraulic structures, such as bridges and their components (i.e., pillars, retaining walls, and railings), from LiDAR datasets collected through field work. This framework is not just a technological advancement but paves the way for critical downstream applications including asset inventory, hydraulic modeling, and safety inspection, showcasing the practical impact of deep learning in enhancing infrastructure management and public safety.

The enhancement in LiDAR data resolution, coupled with the integration of additional sensor data (e.g., RGB), leads to a significant increase in the size of LiDAR datasets, which brings big data challenges to data processing as well as deep learning. A single LiDAR scan can encompass up to billions of points, occupying several gigabytes of storage. This increase in data size further presents substantial challenges for 3D geospatial object detection, especially due to the constraints of computational resources when processing large-scale 3D geospatial datasets. Therefore, current 3D deep learning frameworks, Qi et al. (2017a) for example, used a fixed size of block to partition a large-scale 3D point cloud and followed by a random sampling to structuralize the inferencing datasets. Moreover, some approaches use overlapped blocks to improve model performance. This strategy of data partitioning and sampling can make the input data manageable by limited memory size. Qi et al. (2017a) discusses the challenges of processing large-scale point clouds and the benefits of partitioning them into smaller blocks for memory and computational efficiency. Moreover, data partitioning can also help to capture the local context of the point cloud, especially beneficial to semantic segmentation. Moreover, neural network architectures that leverage Convolutional Neural Network (CNN), as exemplified by Boulch (2020), require structured inputs to facilitate structured convolution operations. This necessity ensures that the input data can undergo convolution in a manner that preserves its inherent structure. Although numerous frameworks, including point-wise MLPs like PointNet and attention-based architectures such as Point Transformer (Zhao et al. 2021), can process point clouds of varying sizes, sampling a fixed number of points often emerges as a practical decision under constraints of computational resources. The configuration of data partitioning and sampling, including block size and the number of points per block, varies depending on the characteristics of the dataset. For instance, Chen (2024) highlighted that the effectiveness of local contextual features is closely related to scale, which directly ties to the settings of data partitioning and sampling. Effective choices of both block size and the number of points to sample can differ based on multiple factors, such as the nature of the dataset, and the demands of the specific application.

However, the strategy of data partitioning and sampling by blocks introduces a significant challenge: not all points from a point cloud may be effectively predicted by deep neural networks. This limitation stems from the inherent design of current deep learning algorithms, which only allow a fixed number of points from a specific spatial extent (i.e., block) to be processed. This problem is commonly overlooked in the studies proposing types of neural network architecture. Furthermore, this aspect of postprocessing is often not explicitly mentioned, being embedded only within the implementation scripts. Therefore, this study aims to delve into the existing solutions to this challenge and seek improved methodologies for handling it, particularly in the context of large-scale 3D point cloud datasets.

In this study, we leveraged the datasets and fine-tuned models from the DeepHyd project (Tang et al. 2022) as a practical study case. Our main contributions are delineated as follows:

1. We explored the current implicit solution to address the issue brought by data partitioning in geospatial object detection for large-scale scenes.
2. We investigated various spatial dependency patterns for different classes using the DeepHyd project as a study case.
3. We implement spatial interpolation methods to refine the post-processing phase of deep learning-based 3D geospatial object detection for large-scale 3D point cloud datasets.

5.2 Literature Review

In practice, those unpredicted points are often labeled based on their nearest neighbor during post-processing, a method depicted in Figure 5.1A and exemplified by (Boulch 2020). Additionally, many studies (Qi et al. 2017a, Wu et al. 2019, Boulch 2020) have employed overlapping spatial extents of blocks as a strategy to mitigate this issue, enabling more points within a specific spatial extent to be predicted and thereby enhancing the accuracy for those unpredicted points. However, relying solely on the nearest neighbor to label unpredicted points is an overly simplistic approach, as a point's label may be influenced not only by its nearest neighbor but also by other surrounding points. Consequently, we have introduced the concept of a surrogate model as a way to improve the accuracy of labeling these unpredicted points.

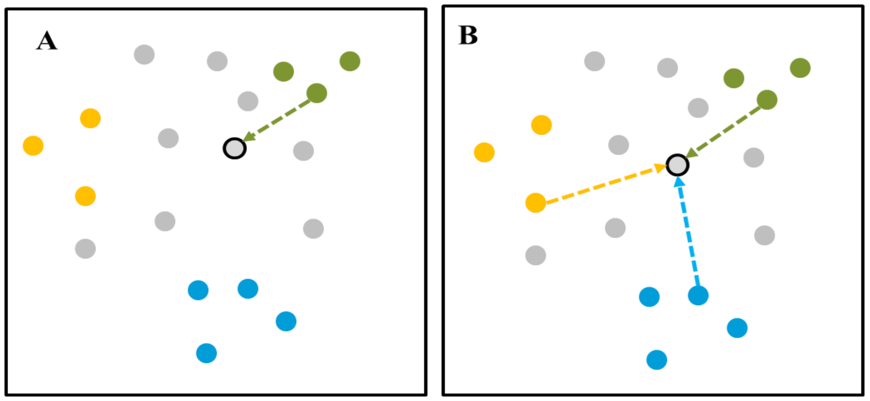


Figure 5.1. Post-processing—assign labels to unpredicted points. Colored points represent predicted points with three different classes (Red, blue, and green). Grey points represent unpredicted points. A: nearest neighbor method; B: spatial interpolation method.

The surrogate model (also known as a metamodel or "model of the model"), as discussed by (Kleijnen 1987, Kleijnen 2009) has been used in engineering to approximate the outcomes of interest, which are either too costly to access, or cannot be directly accessed. Kleijnen (2009) defined that “[m]etamodels are fitted to the I/O data produced by the experiment with the simulation model” (p.). Viewed through this lens, the nearest neighbor method, employed for labeling unpredicted points in 3D deep learning, can be considered a naive form of a surrogate model—a model designed to approximate the outputs generated by primary deep learning models. In this scenario, utilizing the nearest neighbor method as a surrogate during the post-processing phase of 3D semantic segmentation represents a basic approach that could be significantly enhanced through the adoption of more sophisticated surrogate models. Various methods have been used as surrogate modeling techniques in previous studies, such as polynomial response surface (Engelund et al. 1993, Venter, Haftka and Starnes 1996), artificial/deep neural networks (Smith 1993, Tang, Liu and Durlofsky 2020), multivariate adaptive regression splines (Friedman 1991), and radial basis functions (Hardy 1971, Buhmann 2000). For addressing spatial problems specifically (see Figure 5.1B), spatial interpolation methods such as Kriging (Sacks and Welch 1989, Simpson et al. 1998, Kleijnen 2009) and Inverse Distance Weighting (IDW), have been employed due to their capability in spatial interpolation—estimating values at unobserved locations based on observed values from surrounding neighbors. In this light, our study aims to leverage spatial interpolation not as a method for spatial analysis but as a sophisticated surrogate model to address the challenges of 3D deep learning-based geospatial object detection for large-scale scenes.

5.3 Methodology

5.3.1 Metamodel for labeling unpredicted points

The outcome of a deep learning-based 3D geospatial object detection model does not only predict a final label for each predicted point but also evaluates the uncertainty and probability of these predictions, represented as scores and probabilities. Initially, the model outputs a score for each class that it is trained to recognize for a given point. A higher score indicates greater confidence that the point belongs to that class. These scores are subsequently normalized using a softmax function (Goodfellow, Bengio and Courville 2016), converting them into probability values that sum to one, which represent the relative confidence of the model across the different classes. For instance, in a model trained to differentiate between chickens, cats, and dogs, a clear image of a cat might have scores of [50, 80, 30] for chickens, dogs, and cats respectively. These scores could translate to probabilities of [<0.1%, >99.9%, <0.1%] after softmax normalization, demonstrating a high confidence in the cat classification. In contrast, a blurred cat image might result in scores of [5, 8, 3], translating to probabilities of [4.8%, 94.6%, 0.6%], suggesting a high likelihood but less certain of being a cat rather than the other two classes.

Deep learning-based 3D geospatial object detection for large-scene is not only related to predicting labels by the deep neural network itself but also relies on metamodel for labeling all other unpredicted points. It becomes evident that an advanced approach to refining these predictions is essential. This detailed process that models output scores for each class that reflect confidence levels and then normalized to probabilities, lays a crucial foundation for understanding model behavior. For example, the difference in model confidence between clear and blurred images of a cat highlights the variability in certainty that can significantly influence the accuracy and reliability of object detection.

Based upon this understanding, our study pioneered the approach to bring spatial interpolation method, specifically IDW, in addressing the inherent challenges of deep learning-based 3D geospatial object detection for large-scale scenes. Moreover, by manually perturbing the parameter configurations, specifically searching radius and power values, we are able to investigate how the performance of 3D geospatial object detection correlates to the configuration of these parameters, reflecting the underlying spatial dependencies.

5.3.2 Deep learning models of DeepHyd

There are two tasks of DeepHyd software corresponding to two fine-tuned models (see Figure 5.2). One task is to detect bridge from an outdoor scene represented by a 3D point cloud. The other task is to detect different bridge components from the bridge. The project leverages ConvPoint (Boulch 2020) as the architectures and trained models on 41 labeled point cloud datasets for bridges. The models are fine-tuned by detailed hyperparameter tuning.

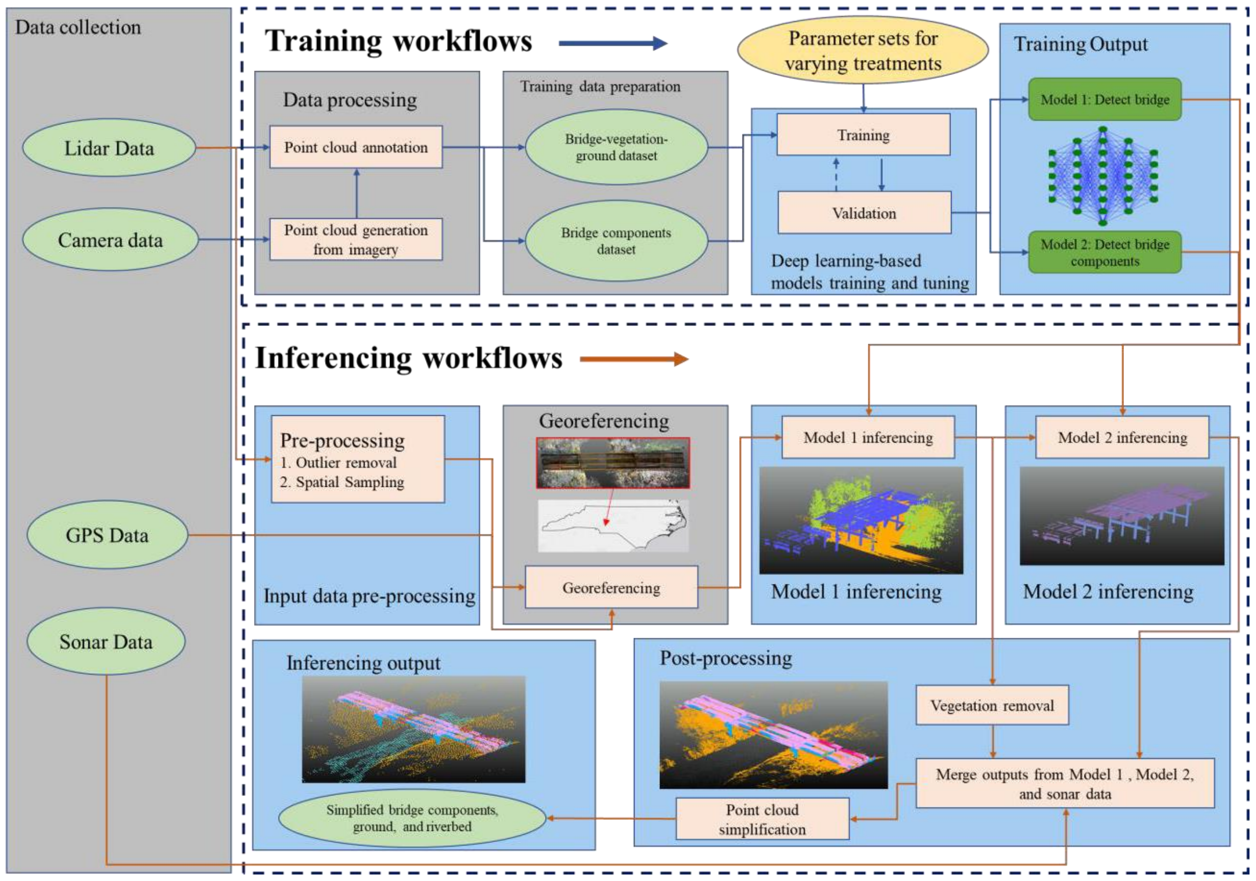


Figure 5.2. Conceptual illustration of the DeepHyd framework and corresponding models (Tang et al. 2022).

ConvPoint (see Figure 5.3) is a pioneering approach in the field of 3D point cloud processing, characterized by its innovative continuous convolutional neural network design specifically tailored for classification and segmentation tasks. Its introduction marked a significant advancement in handling point clouds, which are sets of data points in space representing the external surface of objects or three-dimensional shapes. One of the standout achievements of ConvPoint is its top-ranking performance on the Semantic3D dataset since 2020, where it has consistently outperformed all other cutting-edge neural network architectures entered in the competition[12]. This success underscores the effectiveness of ConvPoint in directly applying

[12] http://www.semantic3d.net/view_results.php?chl=1

convolutions to irregular and unordered point data, especially for large-scale outdoor scenes. By dynamically selecting relevant points within a certain radius for convolution and efficiently managing varying densities and distributions of point clouds, ConvPoint has proven to be a scalable and computationally efficient solution for real-world applications in 3D object detection, classification, and beyond.

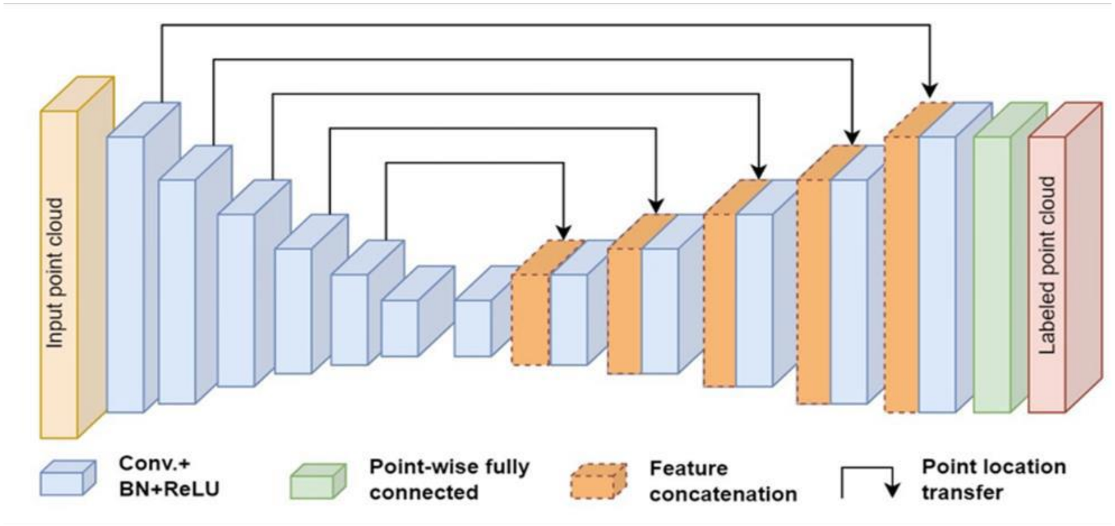


Figure 5.3. Architecture of ConvPoint for 3D object detection (adapted from Boulch, 2020).

Semantic3D, introduced by Hackel et al. (2017), stands as a substantial and diverse dataset specifically tailored for outdoor scene analysis. The Semantic3D dataset offers a detailed and complex dataset with 15 scenes ranging from urban to rural. It covers a wide range of eight semantic categories, man-made and natural terrains, high and low vegetation, structures like buildings and hardscapes (e.g., road light, and fencing), scanning artefacts (e.g., dynamic noise during scanning), and vehicles. The final models of DeepHyd projects leverage a pre-trained model trained on the Semantic3D dataset to boost the performance on detecting bridges and their component structures.

5.4 Data

The data used in this study was collected from selected bridges around Charlotte, NC. We collected in total 10 scans for 7 bridges, where we collect 1-2 scans for each bridge depending on the fieldwork environment. Moreover, we labeled them based on the settings of DeepHyd—one dataset is prepared with labeling bridge, vegetation and ground, and the other one is annotated for bridge components only, such as deck, wall, pillar, and railing (see Figure 5.4 for a demonstration).

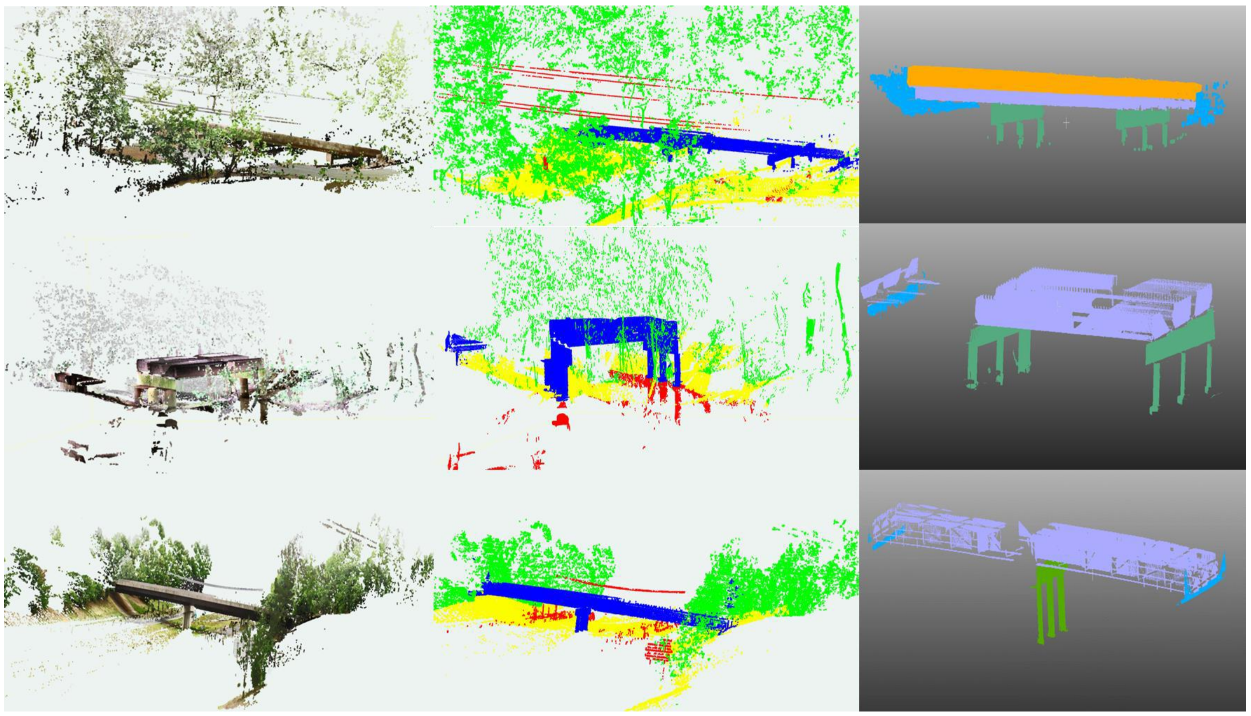


Figure 5.4. Illustration of the collected LiDAR data and corresponding labels.

5.5 Experiment

In this study, we used DeepHyd project as a study case to investigate the effectiveness of using spatial interpolation as the metamodeling for the postprocessing of 3D deep learning-based geospatial object detection. We employed IDW to explore different characteristics of spatial autocorrelations for diverse classes and its impact on the final results. The neighbor searching method can be fixed radius or kNN but we chose to use the latter one. Due to the uneven distribution of 3D point cloud data, particularly in LiDAR-generated point clouds where the density of points increases closer to the scanning center, using fixed-radius settings can lead to empty values for certain points, resulting in labeling issues. However, kNN will not have such an issue and has been broadly used in the 3D deep learning problems. The number of nearest neighbors is set with a minimum value of 1 and the maximum value of 32. The maximum number follows a generic setting (e.g., 16-32 neighbors) when local context is considered for a point during the deep learning process. The range of power is set to from 0.1 to 3.0 with 0.1 as the interval. We assume the maximum value of 3.0 is big enough because of the nature of IDW that the weights of near neighbors as opposed to distant ones increase exponentially with the increase in power. Moreover, our results also support this idea (see response surfaces of performance to kNN and power in Section 5.4). IDW is separately applied to both Model 1 and Model 2 of DeepHyd, where one is to detect bridges on a relative larger scale and the other is to detect bridge components on a relative smaller scale.

The performance measurements used in this study are overall accuracy (OA), mean Intersection of Union (mean IoU), and IoU for each class. IoU and OA are two measurements commonly used for object detection (Rezatofighi et al. 2019, Boulch 2020). IoU has a similar principle as Jaccard index, which is used for similarity measurements in the geography domain (Shelton and Poorthuis 2019). The equations of the three metrics are shown below.

$$IoU = \frac{TP_i}{TP_i + FN_i + FP_i} \tag{5.1}$$

$$mIoU = mean(IoU_i) \tag{5.2}$$

$$OA = \frac{TP_{all}}{TP_{all} + FP_{all}} \tag{5.3}$$

where *TP*, *FP*, and *FN* are True Positive, False Positive, and False Negative; subscript *all* means across all classes and *i* refers to each individual class.

5.6 Results and Discussion

5.6.1 Prediction results on the collected data in this study

In this section, we used the direct predictions results on the dataset collected for this study as a baseline. The direct prediction results are retrieved by directly using the command line-based DeepHyd software[13], where the nearest neighbor is used to label those points that are not predicted by the deep learning-based model in the inferencing step. At first, we demonstrated the number of points for each scan and number of predicted points followed by a predicted rate in Table 1 with respect to Model 1 and Model 2 The predicted rates for various sites depend on the characteristics of the sites (e.g., size of the bridge, and distance of the bridge to the scanning center), the configuration of preprocessing (e.g., sampling), the configuration of the postprocessing (e.g., step size). For Model 1, the predicted rate ranges from 16.16% - 59.58% with an average value of 31.91%. It ranges from 15.70% to 96.71% with an averaged value of 38.46% when it comes to Model 2. We observed the diverse predicted rates with respect to different scans and an averaged predicted rate is less than 40% for both Model 1 and Model 2. In other words, over 60% percent of points are labeled by its nearest neighbor that predicted by the deep neural network. Therefore, it is important to carefully select a method to take care of this post processing step in practice.

Table 5.1. Statistics of number of points and predicted points for each scan.

	Model 1			**Model 2**		
Scan name	**# Points**	**# Predicted**	**Predicted rate**	**# Points**	**# Predicted**	**Predicted rate**
Scan 1	719,106	350,558	48.75%	243,688	160,499	65.86%
Scan 2	1,085,824	242,730	22.35%	766,940	120,383	15.70%

[13] https://rosap.ntl.bts.gov/view/dot/62502

Scan 3	8,009,476	2,539,892	31.71%	3,831,684	1,128,015	29.44%
Scan 4	3,382,589	1,381,216	40.83%	1,060,047	431,250	40.68%
Scan 5	4,239,833	1,289,660	30.42%	1,919,714	451,409	23.51%
Scan 6	1,313,687	782,698	59.58%	272,321	263,374	96.71%
Scan 7	11,061,405	2,790,427	25.23%	1,895,694	638,301	33.67%
Scan 8	6,253,579	1,044,304	16.70%	3,465,144	519,195	14.98%
Scan 9	4,435,961	1,214,976	27.39%	1,222,273	534,298	43.71%
Scan 10	5,705,619	921,896	16.16%	3,202,886	650,979	20.32%
Average	4,620,708	1,255,836	31.91%	1,788,039	489,770	38.46%

We demonstrated the performance measurements in Table 5.2 and Table 5.3 for Model 1 and Model 2 respectively. This performance served as a baseline for comparison purposes. The performance of Model 1 on the collected dataset reaches 86.71% and 73.08% for OA and IoU, which is relatively good for outdoor 3D object detection datasets referring to those model performance in large-scale outdoor classification benchmarks (e.g., Semantic3D). The IoU for bridge is 94.11%, which means the bridge is well detected. These results support the effectiveness of the DeepHyd model to detect bridges from a 3D scene. It appears that vegetation and ground have a relatively lower performance. Tang (2022) also found that vegetation and ground have lower IoU as opposed to the bridge. This can be attributed to the fact that lower vegetation can be potentially misclassified as ground, where DeepHyd software was not designed to identify natural ground or man-made ground (e.g., pavement).

Table 5.2. Performance of Model 1 on the datasets.

Measurement	Values
OA	86.71%
mIoU	73.08%
Bridge IoU	94.11%
Vegetation IoU	51.49%
Ground IoU	73.65%

*OA: Overall Accuracy. mIoU: mean Intersection over Union.

The OA and mIoU for Model 2 are 70.29% and 40.99%, which is relatively lower than that of Model 1. This is due to the limitation of the accessible classification data for Model 2. This can be attributed to many reasons, ranging from the types of bridge, and setting for scanning in the fieldwork (e.g., location and distance to bridge). For example, some bridges may not have specific components, such as pillars, and retaining walls. On the other hand, the setting for scanning may also impact it. For example, under-bridge scans may not capture the railing of a bridge. All the above reasons can lead to an imbalance of the dataset, further resulting in a relatively lower performance of Model 2. Among different bridge components, the deck shows the highest IoU as 78.42% and pillar is ranked in the second with an IoU of 51.95%. Deck and pillar are the two most important classes to be detected for DeepHyd project due to their essential roles in the safety evaluation of the hydraulic structure (Tang, 2022). Improving detection performance in hydraulic modeling is indeed critical for various applications such as hydraulic structure management, and infrastructure planning. Enhanced detection capabilities can lead to more accurate predictions and better-informed decision-making.

Table 5.3. Performance of Model 2 on the datasets.

Measurement	Values
OA	70.29%
mIoU	40.99%
Deck IoU	78.42%
Wall IoU	21.29%
Pillar IoU	51.95%
Railing IoU	12.28%

*OA: Overall Accuracy. mIoU: mean Intersection over Union.

5.6.2 Spatial interpolation results for Model 1

The global performance for Model 1 with IDW for post-processing are demonstrated in Figure 5.5. Since only using the nearest neighbor for post-processing is a special case when we apply IDW with a searching radius as 1 nearest neighbor (i.e., k=1 for kNN). Therefore, the baseline performance is also included in the figure, where it is located on the far-left end of the x axis. The OA and IoU range separately from 86.72% to 86.88%, and from 73.04% to 73.22%. The variations are not obvious (i.e., 0.16% and 0.18% for OA and IoU) but we observe an increasing trend when the number of nearest neighbors increases as suggested by the response surface of Figure 5.5. The increasing pattern along the x axis indicates that the overall performance benefits with more nearest neighbors. Moreover, a lower power results in a higher performance, suggesting the accuracy of points rely more on the local context rather than on only the nearest ones.

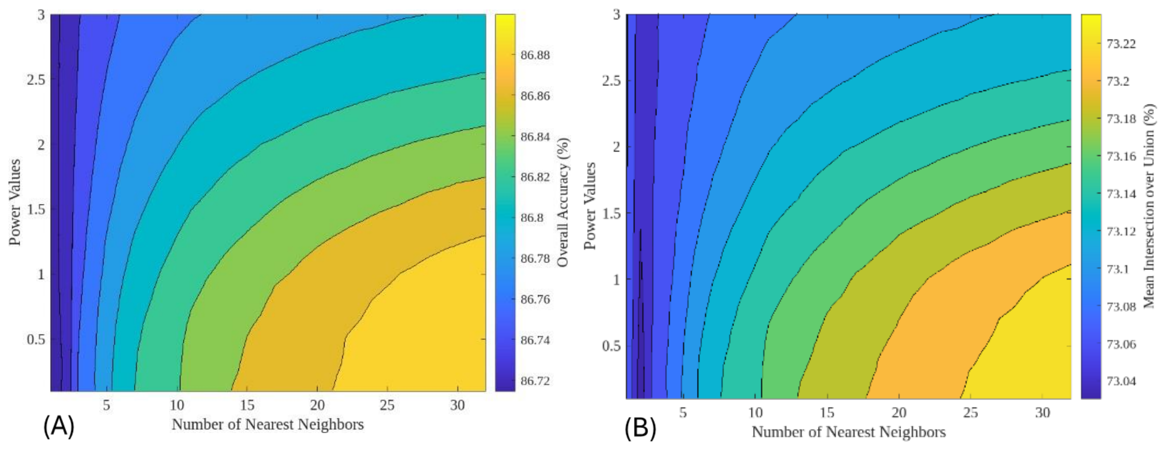


Figure 5.5. Response surface of Overall Accuracy (A) and Intersection over Union (B) for Model 1 with respect to number of nearest neighbor and power of Inverse Distance Weighting.

When it comes to the class-wise performance, we can see diverse patterns for the response surfaces. Generally, there is an increase in IoU of different classes when we apply spatial interpolation, where the increases are separately 0.10%, 0.25%, and 0.45% for bridge, vegetation, and ground, respectively. The IoU of bridge appears to benefit more with a smaller number of nearest neighbors and higher power values (see Figure 5.6A). The pattern of response surface points out that the identification of bridge data depends more on the relatively local neighbors. We can tell that when there is an increase in the number of nearest neighbors, a higher power is more preferred, indicating more weights are given to the near neighbors than distant ones. There is a fluctuation in the trend of the IoU for vegetation along with the increase in the number of nearest neighbor (see Figure 5.6B), where the maximum values are located on the left end of the x axis. Moreover, the IoU is more sensitive to the number of neighbors when the number is around 1 and 2. The Intersection over Union (IoU) metric for ground classification in point cloud data, as indicated by a range from 73.65% to 74.1%, suggests a noteworthy level of accuracy in distinguishing ground points from non-ground points. This range is similar to the pattern observed in the mean Intersection over Union (mIoU) across different classes, implying that the accuracy of point classification is significantly influenced by the local context of points, rather than being solely dependent on the nearest neighbors. This observation underscores the importance of considering a broader neighborhood and the spatial relationships among points to capture the underlying structure and features of the data effectively.

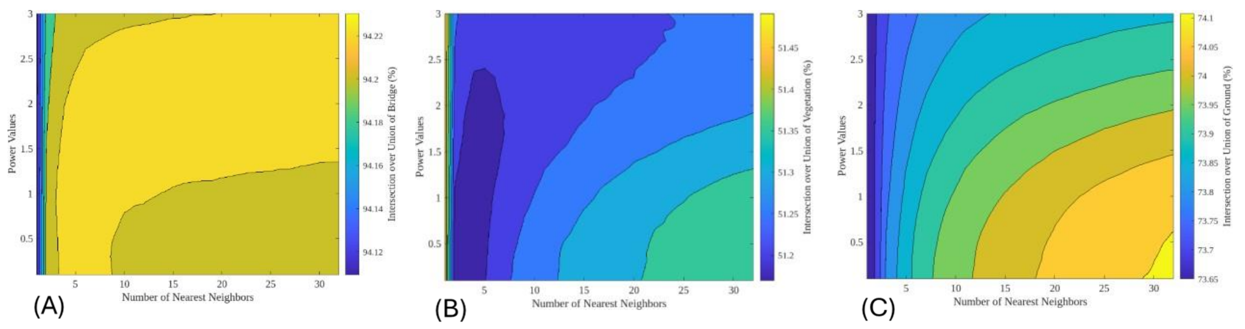


Figure 5.6. Response surface of Intersection over Union on bridge (A), vegetation (B), and ground (C) for Model 1 with respect to number of nearest neighbor and power of Inverse Distance Weighting.

5.6.3 Spatial interpolation results for model 2

The DeepHyd Model 2, specifically designed for detecting bridge components such as decks and pillars from bridge-only point clouds, showcases its effectiveness through its performance metrics. The ranges of OA and mIoU for Model 2 on the dataset collected for this study are separately from 70.3% to 70.9%, and from 41.0% to 42.2%. Compared to Model 1, there is a slightly larger increase when we apply IDW for the post-processing step.

This performance boost aligns with observations from Model 1, suggesting that both models from DeepHyd benefit from considering a larger neighborhood of points. The trend, as depicted in Figure 5.7, illustrates a positive correlation between the model accuracy and the number of neighbors taken into account, indicating that a broader spatial context contributes to the refinement of classification outcomes. Such findings highlight the importance of spatial relationships and the local environment of points in achieving higher accuracy in point cloud segmentation and classification tasks, underscoring the potential of IDW or similar spatial-aware metamodels in elevating the performance of deep learning models like DeepHyd.

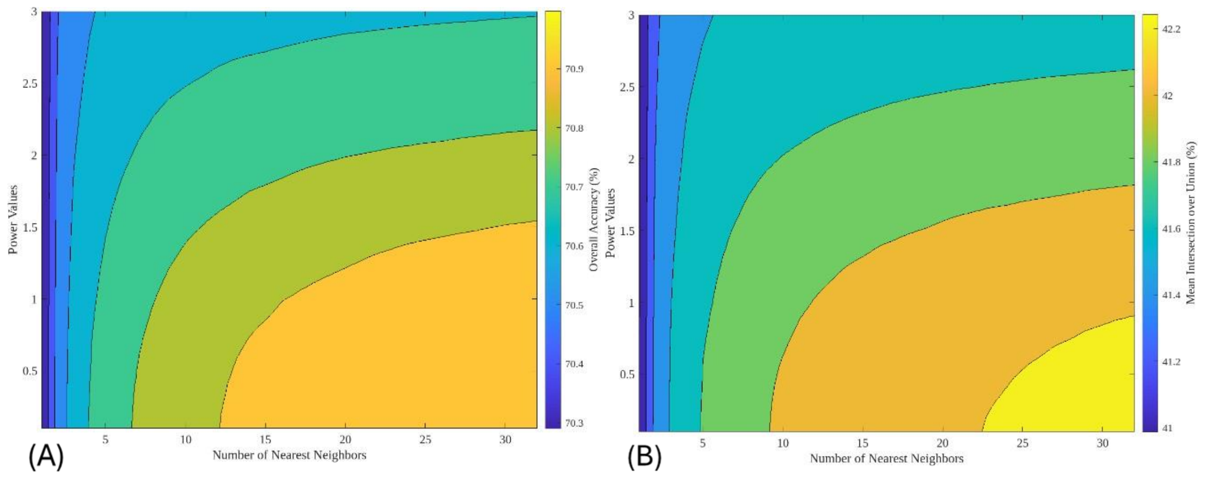


Figure 5.7. Response surface of Overall Accuracy (A) and Intersection over Union (B) for Model 2 with respect to number of nearest neighbor and power of Inverse Distance Weighting.

The improvements observed across various classes in Model 2 provide a compelling insight into the impact of spatial context on point cloud classification. Specifically, the model shows incremental improvements of 0.2% for railing, 0.35% for deck, 1% for wall, and a notable 3.5% improvement for pillar. These enhancements underscore the hypothesis that a broader spatial context significantly contributes to refining classification outcomes for complex structures within bridge-only point clouds.

The optimal performance across different classes in Model 2 (see Figure 5.8) begins to emerge at specific thresholds of nearest neighbors—12 for pillars, 16 for decks, and 29 for retaining walls. Remarkably, beyond these thresholds, the classification accuracy remains stable, demonstrating that additional nearest neighbors do not compromise performance. This finding suggests that while each class requires a minimum spatial context to achieve optimal classification, further expansion of this context does not detract from the accuracy. Such a pattern emphasizes the significance of identifying the minimum number of nearest neighbors necessary to reach peak performance for each class. This strategy not only ensures high precision in classification but also optimizes computational resources by avoiding the inclusion of superfluous neighbors that do not contribute to further accuracy improvements. This approach underlines a nuanced understanding of spatial dependencies within each class, highlighting the delicate balance between the depth of spatial context considered and the computational efficiency in point cloud processing.

Railing presents an interesting case where the relationship between classification accuracy and the number of nearest neighbors deviates from the trend observed in other classes. Its performance shows a negative correlation with an increasing number of neighbors, akin to the behavior noted for vegetation in Model 1. Furthermore, the analysis reveals that railing classifications prefer a higher power value when considering more nearest neighbors, suggesting that railings, similar to vegetation, rely more heavily on immediate spatial neighbors rather than distant ones. This behavior indicates a distinct pattern of spatial autocorrelation for railings, different from other structural components.

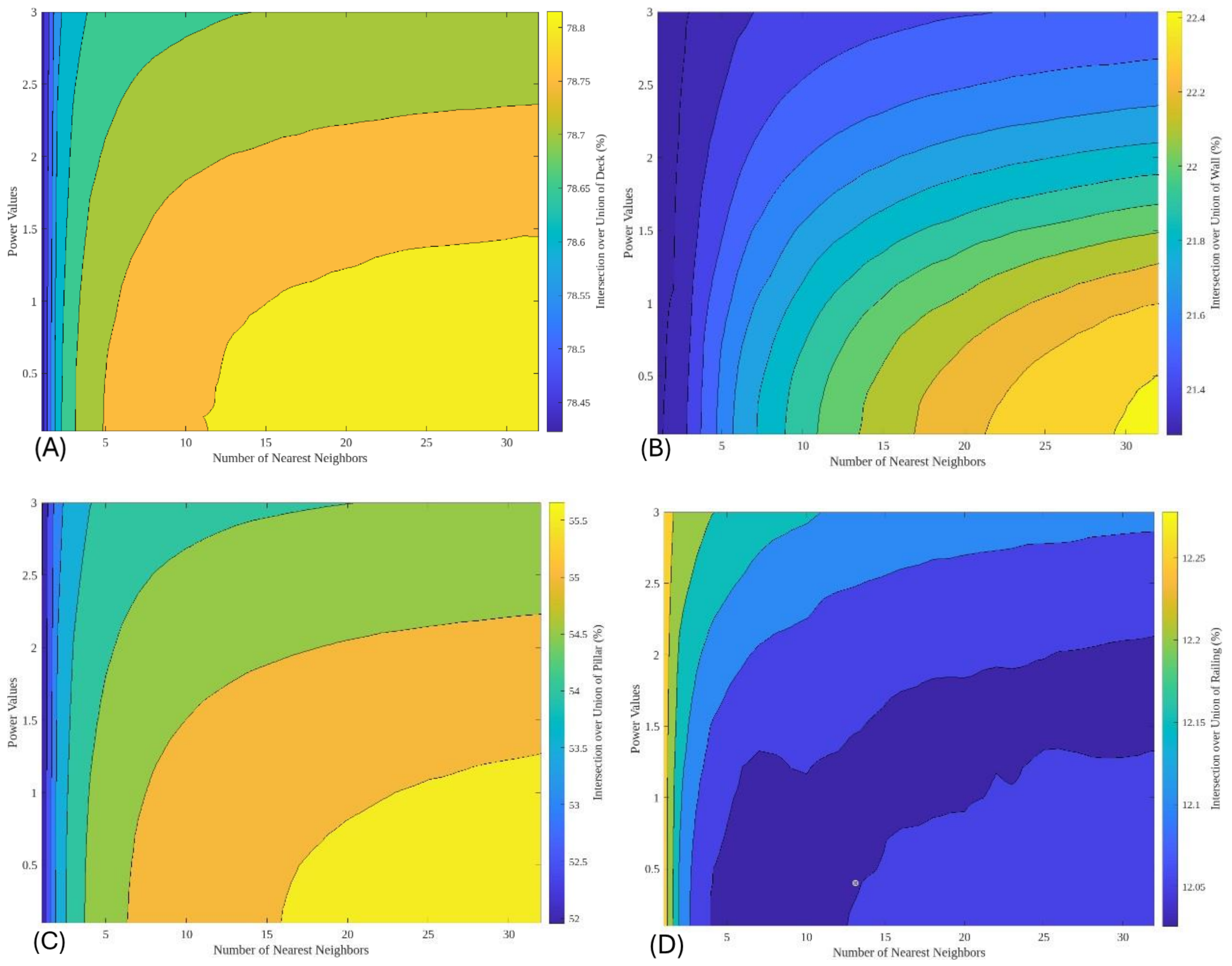


Figure 5.8. Response surface of Intersection over Union on deck (A), retaining wall (B), pillar (C), and railing (D) for Model 2 with respect to number of nearest neighbor and power of Inverse Distance Weighting.

These observations highlight the varying characteristics of spatial autocorrelations within each class, illustrating the complex interplay between spatial context and classification performance in point cloud analysis. Understanding these dynamics is essential for the development of more efficient and accurate point cloud processing algorithms, enabling tailored approaches that consider the unique spatial dependencies of each class to optimize both computational resources and classification accuracy.

In this study, the application of the Inverse Distance Weighting (IDW) method was subject to a uniform parameter setting across all classes, meaning it did not support the customization of the number of points for each class individually. Despite this limitation, the study's findings lay a foundational groundwork for future advancements. By identifying the optimal number of nearest neighbors for peak performance in different classes and highlighting the importance of spatial context, this research paves the road for developing more sophisticated methods. Future approaches could allow for class-specific customization in the number of points considered, potentially enhancing classification accuracy and computational efficiency. This evolution would mark a significant step forward, offering a more tailored and precise approach to point cloud classification that adapts to the unique characteristics and requirements of each class.

5.7 Conclusion

The convergence of advancements in 3D data acquisition technologies and breakthroughs in 3D deep learning has revolutionized the field of 3D geospatial object detection, marking a significant milestone in the digital representation of the physical world. These innovations have not only facilitated the expansion of high-quality (e.g., resolution and additional channels such as RGB) 3D data collections but also underscored the critical need for accuracy in applications impacting public safety and infrastructure management. Within this evolving stage, the DeepHyd project emerges as a notable effort, using deep learning to effectively detect hydraulic structures in large-scale

LiDAR datasets. This study, building upon the foundation laid by the DeepHyd project, sought to address the inherent challenges posed by large-scale 3D point clouds, particularly focusing on the implications of data partitioning and sampling for object detection.

By analyzing datasets optimized for the DeepHyd framework, this study made several important contributions. The study first explicitly investigated the data partitioning method (e.g., using blocks with a fixed size) used to address the challenge brought by the unmanageable size of 3D large-scale point cloud. Furthermore, it delved into the characteristics of spatial dependency patterns for different classes, providing an insight of spatial correlations in point cloud data. Specifically, the study reveals the distinct spatial dependencies of various bridge components, with optimal performance thresholds identified for pillars, decks, and retaining walls, beyond which the additional nearest neighbors does not further improve accuracy. This suggests that there is a maximum spatial extent, while further expansion does not necessarily enhance performance. Such findings emphasize the need for a balanced approach to computational resource allocation and the potential for more sophisticated methods that allow for class-specific customization. The study also highlights the unique spatial autocorrelation patterns within different classes, such as the negative correlation between classification accuracy and the number of nearest neighbors observed in railings, indicating it relies more on nearest neighbors. These observations underscore the complex relations between spatial context and classification performance, pointing to the necessity for approaches that consider unique spatial dependencies for each class.

Reference

Batty, M. (2023) Digital Twins in City Planning. *Nature Computational Science,* 4.

Boulch, A. (2020) ConvPoint: Continuous convolutions for point cloud processing. *Computers & Graphics.,* 88**,** 24-34.

Buhmann, M. D. (2000) Radial basis functions. *Acta Numerica,* 9**,** 1-38.

Chen, T. C. W. T. C. A. S.-E. (2024) Spatial explicit incorporation of spatial autocorrelation features in 3D deep learning [Manuscript submitted for publication]. *Annals of the American Association of Geographers*.

Engelund, W. C., D. O. Stanley, R. A. Lepsch, M. M. McMillin & R. Unal (1993) Aerodynamic configuration design using response surface methodology analysis. *NASA STI/Recon Technical Report A,* 94**,** 10718.

Friedman, J. H. (1991) Multivariate adaptive regression splines. *The Annals of Statistics,* 19**,** 1-67.

Goodchild, M. 2022. The Openshaw effect. 1697-1698. Taylor & Francis.

Goodchild, M. F. (2021) Introduction to urban big data infrastructure. *Urban Informatics***,** 543-545.

Goodfellow, I., Y. Bengio & A. Courville. 2016. *Deep learning*. MIT press.

Hackel, T., N. Savinov, L. Ladicky, J. D. Wegner, K. Schindler & M. Pollefeys (2017) Semantic3D.net: A new Large-scale Point Cloud Classification Benchmark. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences,* IV-1-W1.

Hardy, R. L. (1971) Multiquadric equations of topography and other irregular surfaces. *Journal of geophysical research,* 76**,** 1905-1915.

Kleijnen, J. P. 1987. *Statistical tools for simulation practitioners*. Marcel Dekker.

--- (2009) Kriging metamodeling in simulation: A review. *European Journal of Operational Research,* 192**,** 707-716.

Qi, C., H. Su, K. Mo & L. J. Guibas. 2017a. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 652-660.

Qi, C., L. Yi, H. Su & L. J. Guibas (2017b) Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in Neural Information Processing Systems.*, 30.

Rezatofighi, H., N. Tsoi, J. Gwak, A. Sadeghian, I. Reid & S. Savarese. 2019. Generalized intersection over union: A metric and a loss for bounding box regression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 658-666.

Sacks, W. & T. Welch (1989) Design and analysis of computer experiments. *Statistical Science,* 4**,** 409-435.

Shelton, T. & A. Poorthuis (2019) The nature of neighborhoods: Using big data to rethink the geographies of Atlanta’s neighborhood planning unit system. *Annals of the American Association of Geographers,* 109**,** 1341-1361.

Shi, W. & R. Rajkumar. 2020. Point-gnn: Graph neural network for 3d object detection in a point cloud. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 1711-1719.

Simpson, T., F. Mistree, J. Korte & T. Mauery. 1998. Comparison of response surface and kriging models for multidisciplinary design optimization. In *7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, 4755.

Smith, M. 1993. *Neural networks for statistical modeling*. Thomson Learning.

Tang, M., Y. Liu & L. J. Durlofsky (2020) A deep-learning-based surrogate model for data assimilation in dynamic subsurface flow problems. *Journal of Computational Physics,* 413**,** 109456.

Tang, W., S.-E. Chen, J. Diemer, C. Allan, T. Chen, Z. Slocum, T. Shukla, V. S. Chavan & N. S. Shanmugam. 2022. DeepHyd: A deep learning-based artificial intelligence approach for the automated classification of hydraulic structures from LiDAR and sonar data. North Carolina Department of Transportation. Research and Development Unit.

Venter, G., R. Haftka & J. Starnes, James. 1996. Construction of response surfaces for design optimization applications. In *6th Symposium on Multidisciplinary Analysis and Optimization*, 4040.

Wu, W., Z. Qi & L. Fuxin. 2019. Pointconv: Deep convolutional networks on 3d point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 9621-9630.

Zhao, H., L. Jiang, J. Jia, P. H. Torr & V. Koltun. 2021. Point transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 16259-16268.

6 CONCLUSION

6.1 Summary

This dissertation represents a comprehensive exploration into the realm of GeoAI, specifically focusing on the enhancement of 3D deep learning models' understanding of 3D geospatial data through the lens of spatial principles. The research journey commenced with a foundational understanding of geospatial technologies, including the significance of 3D data acquisition advancements and the evolving role of artificial intelligence in geospatial analytics. At its core, the dissertation seeks to answer pivotal research questions centered around the utility of spatial autocorrelation features in 3D deep learning, the development of a neural network architecture encoding spatial dependencies, and the improvement of post-processing in the task of 3D object detection.

Chapter 3 shed light on the potentials of spatial autocorrelation features, particularly semivariance, in refining 3D deep learning models for geospatial object detection. The incorporation of these features alongside spatial information and color information led to marked improvements in accuracy across varied environmental settings, underscoring the importance of spatial context in enhancing model performance.

Chapter 4 further advanced the discourse by introducing a spatial autocorrelation encoder designed to seamlessly integrate spatial contextual features into 3D deep learning models. This innovation not only boosts accuracy in geospatial object detection but also streamlined the workflow, making sophisticated spatial autocorrelation features more accessible to practitioners without expert knowledge in geospatial analytics.

Chapter 5 focused on a practical study case of 3D geospatial object detection, DeepHyd project which aimed at detecting hydraulic structures within large-scale LiDAR datasets. The chapter delved into the challenges of data partitioning and sampling in large-scale 3D point clouds, offering insights into spatial dependency patterns and the implications for object detection accuracy.

Collectively, this dissertation underscores the pivotal role of spatial autocorrelation features in enhancing the interpretive capabilities of 3D deep learning models. This work contributes to the field of GeoAI by bridging the gap between GIScience and AI, with a focus on offering innovative methodologies for 3D geospatial object detection. It advances theoretical understanding but also has empirical implications in urban planning, environmental monitoring, and beyond, marking a significant step forward in the integration of spatial principles into AI research for enhanced geospatial analytics. This work lays a solid foundation for future research at the intersection of GIScience and AI, driving forward the quest for more intelligent, spatially aware technology solutions.

6.2 Future Work

This dissertation has made efforts in the integration of spatial principles, specifically spatial autocorrelation, with 3D deep learning for enhancing geospatial object detection. The exploration of spatial autocorrelation in 3D deep learning paves roads for future research. This section outlines potential directions for extending this line of study, addressing challenges, and leveraging emerging technologies and methodologies in GIS and AI.

Expanding the Scope of Spatial Features

While this research has focused on the utility of spatial autocorrelation, particularly semivariance of optical data, future studies could explore other spatial and non-spatial features and their integration into deep learning models for a particular study case. For example, spatial context of elevation can be a strong indicator to detect surface microtopography in case study of carbon dynamics. Investigating other aspects of spatial relationships, such as spatial heterogeneity and the scale effect, could lead to further improvements in model performance for geospatial object detection. Additionally, the development of new spatially aware neural network architectures that inherently understand the spatial autocorrelation, and geometry of spatial data could offer novel ways to process and analyze 3D geospatial information.

Leveraging Advanced Deep Learning Architectures

The advancements in deep learning architectures, such as Transformer models and GNNs, provide solid ground for future research. These architectures, known for their ability to capture complex relationships in data, could be adapted and optimized for handling 3D geospatial datasets as well as spatial autocorrelation and spatial heterogeneity nature of them. For example, integrating spatial autocorrelation features within Transformer models designed for 3D data could potentially offer more nuanced understandings of spatial dependencies and enhance object detection capabilities in complex urban and natural environments.

Addressing Challenges Brought by Data Partition and Sampling

As the demand for processing large-scale 3D geospatial datasets, future research can explore approaches to better handle the aforementioned issue brought by the data partitioning and sampling. Exploring efficient ways to handle large-scale 3D point clouds could make deep learning applications more feasible for real-world GIScience problems. Moreover, future studies could explore a broader range of spatial interpolation methods beyond the current implementation. Advanced techniques such as kriging or machine learning-based interpolation could offer better ways to incorporate spatial dependencies, potentially leading to further improvements in detection accuracy across diverse geospatial datasets.

Ethical Considerations and Societal Impact

Future studies could extend the methodologies developed in this dissertation to practical applications such as urban planning, environmental monitoring, and autonomous navigation. Each of these domains presents unique challenges and opportunities for leveraging spatial principles and deep learning to derive insights from 3D geospatial data. As the capabilities of AI in understanding and interpreting 3D geospatial data advance, it is crucial to consider the ethical implications and societal impacts of these technologies. Future research should not only focus on technical advancements but also address concerns related to privacy, data security, and the potential bias in AI algorithms. Developing frameworks and guidelines for the responsible use of 3D deep learning in GIScience could ensure that these technologies are safe and trustworthy to the public.

This dissertation represents a step forward in the integration of spatial principles into 3D deep learning for geospatial object detection. As the evolution of GeoAI, these future studies will play an essential role in shaping the next generation of geospatial technologies.

Reference

Analytics, M. (2020) Global survey: The state of AI in 2020. *Image*.

Atzori, L., A. Iera & G. Morabito (2017) Understanding the Internet of Things: definition, potentials, and societal role of a fast evolving paradigm. *Ad Hoc Networks,* 56**,** 122-140.

Bengio, Y., Y. Lecun & G. Hinton (2021) Deep learning for AI. *Communications of the Association for Computing Machinery,* 64**,** 58-65.

Couclelis, H. (1986) Artificial intelligence in geography: Conjectures on the shape of things to come. *The Professional Geographer,* 38**,** 1-11.

De Deuge, M., A. Quadros, C. Hung & B. Douillard. 2013. Unsupervised feature learning for classification of outdoor 3D scans. In *Australasian Conference on Robitics and Automation*. araa.asn.au.

Goodchild, M. (2010) Twenty years of progress: GIScience in 2010. *Journal of Spatial Information Science***,** 3-20.

Goodchild, M. F. & W. Li (2021) Replication across space and time must be weak in the social and environmental sciences. *Proceedings of the National Academy of Sciences of the United States of America,* 118.

Hackel, T., N. Savinov, L. Ladicky, J. D. Wegner, K. Schindler & M. Pollefeys (2017) Semantic3d. net: A new large-scale point cloud classification benchmark. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences,* IV-1/W1**,** 91–98.

Han, Z., Z. Liu, J. Han, C.-M. Vong, S. Bu & C. L. P. Chen (2016) Mesh convolutional restricted Boltzmann machines for unsupervised learning of features with structure preservation on 3-D meshes. *IEEE Transactions on Neural Networks and Learning Systems,* 28**,** 2268-2281.

Hinton, G. (2009) Deep belief networks. *Scholarpedia,* 4**,** 5947.

Ioannidou, A., E. Chatzilari, S. Nikolopoulos & I. Kompatsiaris (2017) Deep Learning Advances in Computer Vision with 3D Data: A Survey. *Association for Computing Machinery Computing Surveys,* 50**,** 1-38.

LeCun, Y., Y. Bengio & G. Hinton (2015) Deep learning. *Nature,* 521**,** 436-444.

Li, W. (2021) GeoAI and Deep Learning. *The International Encyclopedia of Geography.***,** 1-6.

Liu, M., M. Salzmann & X. He. 2014. Discrete-continuous depth estimation from a single image. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 716-723. IEEE.

Magomadov, V. S. (2020) The Industrial Internet of Things as one of the main drivers of Industry 4.0. *Institute of Physics Conference Series: Materials Science and Engineering,* 862**,** 032101.

Maturana, D. & S. 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)*, 922-928. ieeexplore.ieee.org.

Minaee, S., N. Kalchbrenner, E. Cambria, N. Nikzad, M. Chenaghlu & J. Gao (2021) Deep Learning-10based text classification: A comprehensive review. *Association for Computing Machinery Computing Surveys,* 54**,** 1-40.

Nassif, A. B., I. Shahin, I. Attili, M. Azzeh & K. Shaalan (2019) Speech Recognition Using Deep Neural Networks: A Systematic Review. *IEEE Access,* 7**,** 19143-19165.

Openshaw, S. (1992) Some suggestions concerning the development of artificial intelligence tools for spatial modelling and analysis in GIS. *The Annals of Regional Science,* 26**,** 35-51.

Openshaw, S. & C. Openshaw. 1997. *Artificial intelligence in geography*. John Wiley & Sons, Inc.

Philbeck, T. & N. Davis (2018) The fourth industrial revolution. *Journal of International Affairs,* 72**,** 17-22.

Qi, C., H. Su, K. Mo & L. J. Guibas. 2017. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 652-660.

Reichstein, M., G. Camps-Valls, B. Stevens, M. Jung, J. Denzler, N. Carvalhais & Prabhat (2019) Deep learning and process understanding for data-driven Earth system science. *Nature,* 566**,** 195-204.

Rottensteiner, F., G. Sohn, M. Gerke, J. D. Wegner, U. Breitkopf & J. Jung (2014) Results of the ISPRS benchmark on urban object detection and 3D building reconstruction. *ISPRS Journal of Photogrammetry and Remote Sensing,* 93**,** 256-271.

Schwab, K. 2017. *The fourth industrial revolution*. Crown.

Serna, A., B. Marcotegui, F. Goulette & J.-E. Deschaud. 2014. Paris-rue-Madame database: A 3D mobile laser scanner dataset for benchmarking urban detection, segmentation and classification methods. In *4th International Conference on Pattern Recognition, Applications and Methods ICPRAM 2014*. hal.science.

Silberman, N., D. Hoiem, P. Kohli & R. Fergus. 2012. Indoor segmentation and support inference from RGBD images. In *European Conference on Computer Vision (5),* 746-760. Florence, Italy.

Silver, D., A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel & D. Hassabis (2016) Mastering the game of Go with deep neural networks and tree search. *Nature,* 529**,** 484-489.

Singh, T., S. Sharma & S. Nagesh (2017) Socio-economic status scales updated for 2017. *Medical Science Monitor,* 5**,** 3264.

Smith, T. R. (1984) Artificial intelligence and its applicability to geographical problem solving. *The Professional Geographer,* 36**,** 147-158.

Song, S., S. P. Lichtenberg & J. Xiao. 2015. SUN RGB-D: A RGB-D scene understanding benchmark suite. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* 567-576. IEEE.

Su, H., S. Maji, E. Kalogerakis & E. Learned-Miller. 2015. Multi-view convolutional neural networks for 3D shape recognition. In *IEEE International Conference on Computer Vision*, 945-953. Santiago, Chile.

Tuan, Y.-F. 1977. *Space and place: The perspective of experience*. University of Minnesota Press.

Vallet, B., M. Brédif, A. Serna, B. Marcotegui & N. Paparoditis (2015) TerraMobilita/iQmulus urban point cloud analysis benchmark. *Computers & Graphics,* 49**,** 126-133.

VoPham, T., J. E. Hart, F. Laden & Y.-Y. Chiang (2018) Emerging trends in geospatial artificial intelligence (geoAI): Potential applications for environmental epidemiology. *Environmental Health,* 17**,** 1-6.

Wu, Z., S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang & J. Xiao. 2014. 3D ShapeNets: A deep representation for volumetric shapes. In *IEEE Conference on Computer Vision and Pattern Recognition*. Boston, Massachusetts.

Young, T., D. Hazarika, S. Poria & E. Cambria (2018) Recent trends in deep learning based natural language processing. *IEEE Computational Intelligence Magazine,* 13**,** 55-75.

Yusoff, I. M., A. Ramli & N. A. M. Al-Kasirah (2021) Geospatial data and technology application towards managing flood disaster in the context of industrial revolution 4.0 (IR4.0). *Journal of Agriculture, Science And Technology,* 1**,** 38-69.

Zhu, Z., P. Luo, X. Wang & X. Tang (2014) Multi-view perceptron: A deep model for learning face identity and view representations. *Advances in Neural Information Processing Systems*, 27.

Zlatanova, S., A. Rahman & M. Pilouk (2002) 3D GIS: Current status and perspectives. *International Archives of Photogrammetry Remote Sensing and Spatial Information Sciences*, 34**,** 66-71.

Zlochower, A., D. S. Chow, P. Chang, D. Khatri, J. A. Boockvar & C. G. Filippi (2020) Deep learning AI applications in the imaging of glioma. *Topics in Magnetic Resonance Imaging*, 29**,** 115-110.