sensors

MDPI

Article

A Spatial Adaptive Algorithm Framework for Building Pattern Recognition Using Graph Convolutional Networks

Weijia Bei [1], Mingqiang Guo [1,*] and Ying Huang [2]

[1] School of Geography and Information Engineering, China University of Geosciences (Wuhan), Wuhan 430074, China; 20171000718@cug.edu.cn

[2] Wuhan Zondy Cyber Technology Ltd. Co., Wuhan 430074, China; huangying@mapgis.com

* Correspondence: gmqandjxs@163.com; Tel.: +86-13207161520

Received: 19 October 2019; Accepted: 11 December 2019; Published: 13 December 2019

Abstract: Graph learning methods, especially graph convolutional networks, have been investigated for their potential applicability in many fields of study based on topological data. Their topological data processing capabilities have proven to be powerful. However, the relationships among separate entities include not only topological adjacency, but also correlation in vision, for example, the spatial vector data of buildings. In this study, we propose a spatial adaptive algorithm framework with a data-driven design to accomplish building group division and building group pattern recognition tasks, which is not sensitive to the difference in the spatial distribution of the buildings in various geographical regions. In addition, the algorithm framework has a multi-stage design, and processes the building group data from whole to parts, since the objective is closely related to multi-object detection on topological data. By using the graph convolution method and a deep neural network (DNN), the multitask model in this study can learn human thoughts through supervised training, and the whole process only depends upon the descriptive vector data of buildings without any ancillary data for building group partition. Experiments confirmed that the method for expressing buildings and the effect of the algorithm framework proposed are satisfactory. In summary, using deep learning methods to complete the tasks of building group division and building group pattern recognition is potentially effective, and the algorithm framework is worth further research.

Keywords: building pattern; node classification; graph partition; graph convolutional networks; random forest; graph convolutional neural networks; machine learning

1. Introduction

Buildings are important entities in the fields of city computing and city perception. The distributive characteristics of different building groups can be visually summarized as patterns considered as the fine-grained features of the city. In addition, patterns of building groups play an important role in map generalization and navigation [1,2], and the indices (e.g., the area, perimeter, orientation and compactness) of buildings are descriptive enough for deep learning methods to accomplish some classical tasks of building pattern classification [3]. In general, building patterns can be divided into regular patterns and irregular patterns. Grid-like patterns are the main manifestation of regular patterns, while irregular patterns mainly consist of I-shape, L-shape and independent types [4].

Existing methods for building pattern recognition usually partition building groups with the help of road networks and other ancillary data [5–7], and thus the applicability in some geographic analysis scenarios is weakened [8]. Therefore, proposing a building group partition method with high accuracy is essential for the data independence of finishing the building pattern recognition task.

The building pattern recognition task is equivalent to the multi-object recognition task, and the difference is that the former is based on topological data, compared to the traditional multi-object recognition task of computer vision. The building group partition operation in the building pattern recognition process is like region proposal methods. Fast R-CNN [9] and its derivative algorithms are a kind of effective, multi-object recognition framework [10,11]. The contour of the target object is often irregular. Hence convolutional neural networks (CNNs) are used to calculate the probability of whether the proposal region contains target objects [12]. On the basis of Fast R-CNN, the appearance of Mask R-CNN brought the research to the stage of instance segmentation [13]. Mask R-CNN is also based on region proposal methods and identifying each pixel as the background or as a part of the object by a fully convolutional network (FCN) [14]. To identify all building patterns in a building group, the instance segmentation idea is transferable. Inspired by the usage of the FCN in Mask R-CNN, in this study we summarize three spatial states of one building: the edge state, the inner state and the free state. Specifically, there is an analogy between the buildings of the edge state and the contours of an object output by Mask R-CNN, and the method proposed in this study classifies each building into one of the three states mentioned above by using the graph convolutional network (GCN) model, like the usage of the FCN.

Graph neural networks (GNNs) and GCNs exhibit excellent performance based on topological data in different research fields, including social networks [15], protein interface prediction [16], disease prediction [17,18] and remote sensing image processing [19]. GNNs and GCNs accomplish information aggregation according to the adjacent relations between nodes in a graph, aiming to perceive the topological features of different nodes. This notion is closely related to spatial association [20] and the first law of geography [21] because of the natural formation of adjacent relations based on distances. However, in general circumstances, the relationships between objects not only include topological adjacency but also shape similarity, especially in research on building pattern recognition [3,22]. The graph learning methods mentioned above are normally applied to non-Euclidean data, but seldom focus on the visual characteristics and the spatial distribution of the nodes [23].

To perceive the relationships among the graph node and its neighbors, graph embedding is employed, which learns to represent graph nodes with n-dimensional vectors. Graph embedding has a close connection with methods of the representation-based classification (e.g., some up-to-date works, such as LMRKNN [24], TPCRC [25], the novel DCRC method via l2 regularizations [26] and MKFLE [27]). Inspired by the representation learning, graph embedding methods on graph domain (e.g., DeepWalk [28], node2vec [29], LINE [30] and SDNE [31]) and some specific methods like those seen in [32] and [15] are proposed, which accomplish information aggregation based on the adjacent relationships among nodes on graphs. The similarities and the discriminations of the graph nodes are reflected by the representation vectors, with the expectations that the nodes can be correctly classified into their own classes.

In this work, we propose a representation method, named as the shifting degree of adjacency weight, to describe the spatial correlation between the buildings and the visual characteristics of the building nodes. To avoid the overfitting problem [33], the method is rotate-invariant and shift-invariant. Besides, in order to achieve the objective of building pattern recognition, an algorithm framework is proposed in this study. The framework has a multi-stage design and processes the building group data from whole to parts, since the building pattern extraction during the workflow is associated with the multi-object detection on graphs. Additionally, the mentioned graph learning methods encode the nodes by the information passing through and aggregation between adjacent nodes to derive the features of nodes, which solves the problems of the unfixed size of vertices and the uncertain adjacent relationships. In this study, a novel graph convolution operation is introduced for better performance of feature encoding.

With a symmetric normalized Laplacian matrix, the adjacency information of nodes has been normalized during the aggregation operation to address overfitting [15]. In addition, local weight sharing is generalized to graph structure through a polynomial approximation of the Laplacian matrix [32,34]. In addition, a graph convolutional neural network (GCNN) [35,36] is constructed by combining the graph convolution operation with a deep neural network for the graph representation learning related to the building pattern recognition task. Generally, the whole process only depends upon the descriptive vector data of buildings without any ancillary data for building group partition, which improves the applicability of the proposed method.

This study focuses on developing an algorithm framework to finish building group partition and pattern recognition (e.g., I-shaped, L-shaped, grid-like and single type pattern) tasks based only on the vector data of building contours. In addition, experiments for exploring reasonable model structures have been conducted for a satisfactory and convincing result.

The remainder of the paper is structured as follows: Section 2 introduces the experimental datasets for the model training and testing. Section 3 describes the principles of the proposed methods in details, and Section 4 articulates the procedure of the algorithm framework. The experiments and results are presented in Section 5, and some issues are discussed in Section 6. Finally, Section 7 summarizes this research and presents the future works.

2. Study Materials

Beijing's Xicheng District and the core areas of the city of Xi'an (Figure 1) were selected as the study regions. The selection is reasonable because of their long history and their important development positions. The two study regions include various stages of urban development in China, and therefore the building distributions and contours are in accordance with the multiplicity principle. The experimental datasets contain the vector data of building contours at a scale of 1:2000 in the two mentioned areas for 2017. The vector data of each building contour consists of a series of key points recording longitude and latitude data. In the annotation process, we first selected the data of building contours in random rectangular areas as separate data blocks, with 75–154 buildings per block. The models used by the proposed algorithm framework are trained with supervised or semi-supervised learning, and the datasets for the training and testing of different models are labeled according to the specific tasks. In general, three datasets were constructed for the three tasks: building state identification, building node clustering and building pattern recognition.

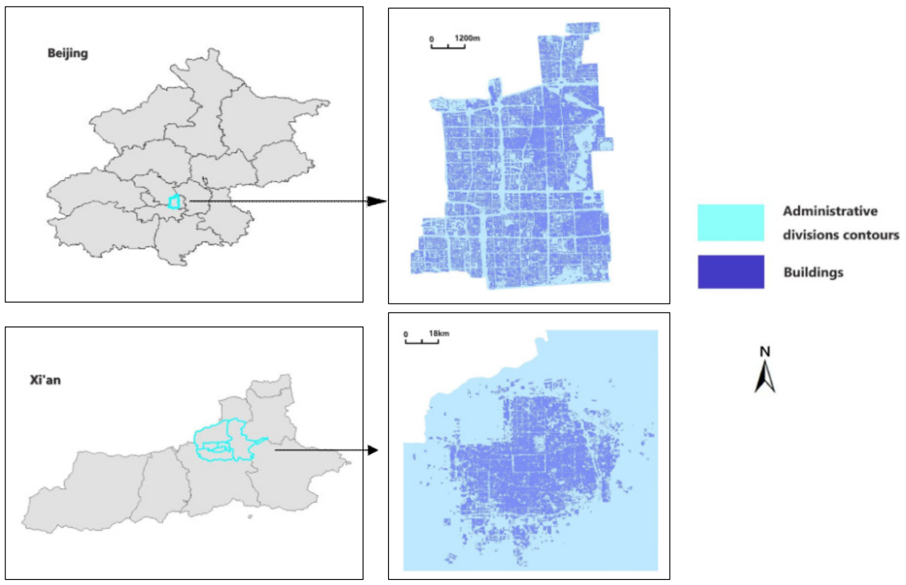

Figure 1. Experimental datasets.

3. Methodology

On the basis of the adjacency relationships among the building nodes, each data block, which is referring to a building group, is processed from whole to parts in the proposed framework with a multi-stage design. All the building nodes are firstly classified into three spatial states for further processes. The four major parts, building node state identification, building group partition, fine-grained partition for building blocks and building pattern recognition, are explained with details in the following sections.

3.1. Building Node State Identification

3.1.1. Definition of Three Building Node States

According to the possible spatial correlation between the building nodes, three kinds of building node states are defined:

- Edge state. Intuitively, the edge state buildings are located on the edge of a building block. Their unique characteristic is that the contrast between the buildings of their two sides is strong (e.g., the bright yellow buildings shown in Figure 2). The contrast embodied by the difference of the descriptive vectors of building nodes will lead to unique feature encoding through the graph convolution operation. Therefore, the definition of edge state is reasonable, and it is indicative of the inner state building.
- Inner state. Buildings located in the same building pattern are similar in terms of size, outline and spatial position; hence, buildings located in the same pattern are defined as the inner state buildings (e.g., the blue buildings shown in Figure 2).
- Free state. Normally, there is one building that exhibits independence because of its spatial distance among others. We define such buildings as free state buildings (e.g., the orange buildings shown in Figure 2).

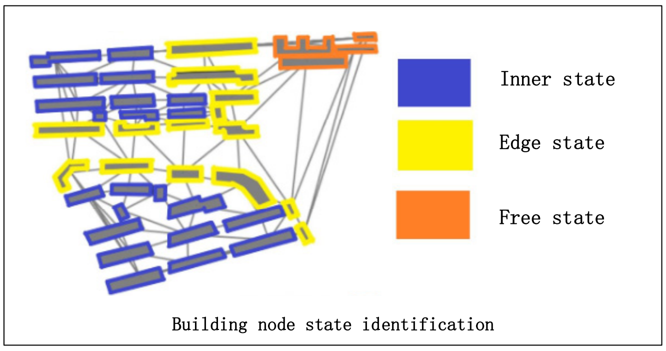

Figure 2. Definition of three building node states.

3.1.2. Descriptive Methods for Building Features

We can quantify the differences stated in Section 3.1.2 through descriptive vectors constructed by the variables summarized in Table 1. The definitions of shifting degree of adjacency weight and orientation are in Sections 3.1.2.1 and 3.1.2.2, respectively.

Table 1. Description of spatial characteristics of individual buildings along with their equations and short descriptions.

Variable	Index	Equation	Description
Position feature	Shifting degree of adjacency weight in width direction	-	See Section 3.1.2.1
	Shifting degree of adjacency weight in height direction	-	See Section 3.1.2.1
Size	Area index	$A_b/\max(A_b)$	Building area with normalizing operation
	Perimeter index	$P_b/\max(P_b)$	Building perimeter with normalizing operation
Orientation	Orientation index	-	See Section 3.1.2.2
Shape	Compactness	$\frac{4\pi A_b}{P_b^2}$	Quadratic relationship between the area and the perimeter [37]
	Concavity	$\frac{A_b}{A_{ch}}$	Area ratio of the building to its convex hull [37]

3.1.2.1. Definition of the Shifting Degree of Adjacency Weight

The spatial distribution of buildings is fundamental for building group partition. In a building group, it is intuitive to treat buildings that are close to each other as one building block. The sparse part reflects the boundary between two separate partitions. Therefore, the distance between buildings can be used to describe the sparse part [4]. However, distance is merely one descriptive parameter in one-dimensional space, and it is not enough to describe the spatial distribution of the buildings. Thus, we need two-dimensional indices to express the sparsity or the density of the buildings in two-dimensional space (see Figure 3a,c), and the shifting degree of adjacency weight is defined in this study.

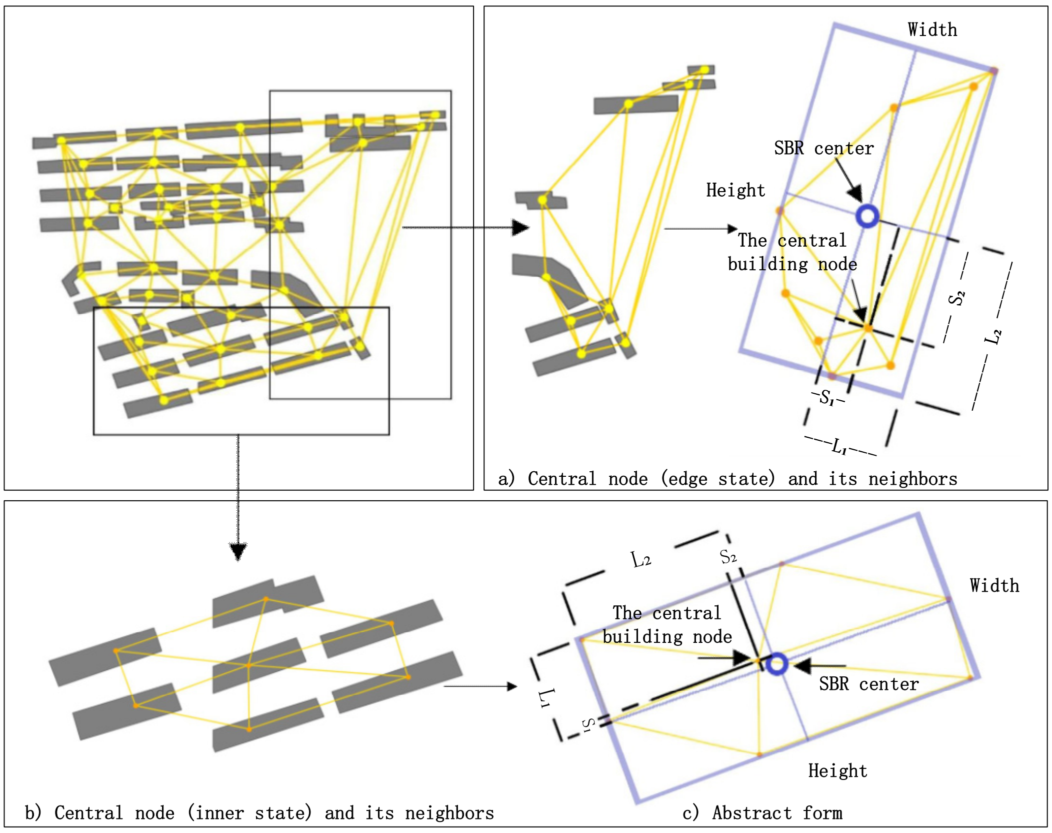
Width
SBR center
Height
The central building node
S_2
L_2
S_1
L_1
a) Central node (edge state) and its neighbors
L_2
S_2
The central building node
Width
SBR center
L_1
S_1
Height
b) Central node (inner state) and its neighbors
c) Abstract form

Figure 3. Descriptions of shifting degree of x–y-dimension weight. (**a**) Central node (edge state) and its neighbors. (**b**) Central node (inner state) and its neighbors. (**c**) Abstract form.

As shown in Figure 3a, we first calculate the center coordinate, width and length of the smallest bounding rectangle (SBR) of the central building node and its neighbors. Specifically, (see Figure 3a,c), the width is the edge of the SBR that is parallel to the X axis with counterclockwise rotation of the smallest degree. The adjacent edge is the height of the SBR. The two-dimensional indices are given by

$$D_w = \frac{2S_1}{L_1} \tag{1}$$

$$D_h = \frac{2S_2}{L_2} \tag{2}$$

where D_w denotes the shifting degree of adjacency weight in the width direction and D_h denotes the shifting degree of adjacency weight in the height direction. The geometrical meanings of S_1, S_2, L_1, and L_2 are shown in Figure 3a,c where L_1 denotes the half of the width and L_2 denotes the half of the height. S_1 and S_2 mean the offset distances between the central building node and the center coordinate of the SBR in the width direction and height direction, respectively.

3.1.2.2. Description for Building Orientation

The difference in building orientation is important for judging whether buildings should be in the same building pattern. As shown in Figure 4, we derive the angle $N°(N° \in [0, \pi])$ between the width of the SBR (see Section 3.1.2.1) and the X axis. The expression for the descriptive variable is

$$O = \begin{cases} \frac{90-N}{180}, & W > H, \\ \frac{180-N}{180}, & H > W \end{cases} \tag{3}$$

where H and W are the height and width of the SBR, respectively, as defined in Section 3.1.2.1. The output value of the expression above is normalized during the calculation procedure to avoid the overfitting problem [33] during the training process.

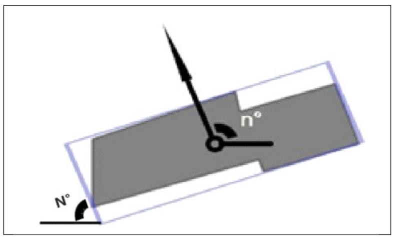

Figure 4. Description of the building orientation.

3.1.3. Graph Convolutional Network

The building group partition and building pattern recognition are based on the feature encoding of the building nodes in this study. The descriptive indices for buildings are given in Table 1. Similar to the human visual system, the GCN model makes judgments on the basis of the differences among the building and its Kth-order neighbors, as human's eyes distinguish detail based on the gradient information of pixels.

As shown in Figure 3b, the shifting degree of adjacency weight (Section 3.1.2.1) is small when the building is located in the inner building group, while the shifting degree is relatively larger when the building is located on the edge of the building block (Figure 3a). This is one of the differences between the buildings in various states. In addition, differences are also embodied in the aspects of size, shape, orientation and other indices. The model learns the judgment rules by using the training samples.

On the basis of the concepts above, the process for deriving building node encoding is as follows: First, we consider the situation of one-dimensional linear adjacency. As shown in Figure 5, only building T_{i-1} and building T_{i+1} are adjacent to building T_i. Therefore, the difference information δ_i is derived from the following aggregation operation:

$$\delta_i = (\varphi_i - \varphi_{i+1}) + (\varphi_i - \varphi_{i-1}) \tag{4}$$

where φ_i denotes the descriptive vector of building T_i. One sample of a real building distribution shown in Figure 6. Similar to Equation (4), the aggregation operation is given by

$$\delta_i = \sum_{j=1}^{N} A_{i,j} \times \left(\varphi_i - \varphi_j\right) = \varphi_i \deg(i) - \sum_{j=1}^{N} A_{i,j}\varphi_j \tag{5}$$

where A refers to the adjacency matrix and $\deg(i)$ denotes $\sum_j A_{i,j}$.

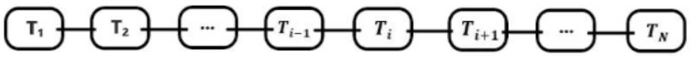

Figure 5. One-dimensional linear adjacency.

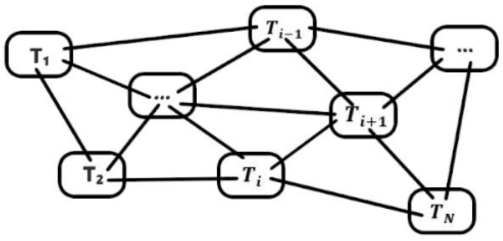

Figure 6. Sample of a real building distribution.

For each building node in a graph, the computation process can be described based on the matrix operation

$$\begin{bmatrix} \delta_1 \\ \vdots \\ \delta_N \end{bmatrix} = \begin{bmatrix} \deg(1)\varphi_1 \\ \vdots \\ \deg(\mathrm{N})\varphi_N \end{bmatrix} - A \begin{bmatrix} \varphi_1 \\ \vdots \\ \varphi_N \end{bmatrix} \tag{6}$$

We define vectors $\varphi = [\varphi_1, \varphi_2, \ldots, \varphi_N]^T$ and $\delta = [\delta_1, \ldots, \delta_N]$ and get

$$\delta = D\varphi - A\varphi = (D - A)\varphi = L\varphi \tag{7}$$

where L is the Laplacian matrix. Equation (7) shows that the usage of the Laplacian matrix is equivalent to the aggregation operation. A symmetric normalization operation for the Laplacian matrix [15] is implemented to address overfitting. The expression is

$$L^{sys} = D^{-\frac{1}{2}} L D^{-\frac{1}{2}} = I_N - D^{-\frac{1}{2}} A D^{-\frac{1}{2}} \tag{8}$$

where I_N is the identity matrix of size N.

The Fourier transform is an effective tool in the fields of signal analysis and image processing; it converts the original signal or image information into the Fourier domain [3,38]. In this study, we first extract the adjacency information from L^{sys} (Equation (8)) by using the graph Fourier transform, and then we introduce a polynomial approximation operation implemented on the modified Laplacian matrix that optimizes the computational procedure.

First, the spectral decomposition for the Laplacian matrix is given by

$$L = U\begin{bmatrix} \lambda_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \lambda_N \end{bmatrix}U^{-1} \tag{9}$$

where $U = \left(\vec{u_1}, \vec{u_2}, \ldots, \vec{u_N}\right)$ and λ_n $(n \in [0, N-1])$ is the nth eigenvalue of the Laplacian matrix. Because U is an orthogonal matrix, and thus $U^T = U^{-1}$, according to the definition of the Fourier transform, the graph Fourier transform [3] is

$$F(\lambda_l) = \hat{f}(\lambda_l) = \sum_{i=1}^{N} \chi_l^T(i) f_l(i) \tag{10}$$

where f_l refers to the signal (the input vector) and $\{\chi_l\}_{l=0}^{N-1}$ are the eigenvectors of the Laplacian matrix. The computing process in detail is

$$\begin{bmatrix} \hat{f}(\lambda_1) \\ \vdots \\ \hat{f}(\lambda_N) \end{bmatrix} = \begin{bmatrix} \chi_1(1) & \cdots & \chi_1(N) \\ \vdots & \ddots & \vdots \\ \chi_N(1) & \cdots & \chi_N(N) \end{bmatrix} \begin{bmatrix} \vec{f_1} \\ \vdots \\ \vec{f_N} \end{bmatrix} \tag{11}$$

The inverse Fourier transform is defined as $f_l = \sum_{i=0}^{N-1} \hat{f}(\lambda_l)\chi_l$. On the basis of the derivation above, the convolution can be first converted into a point-wise product in the Fourier domain, and then reconverted into the vertex domain [3] as follows:

$$f * g = \sum_{l=0}^{N-1} \hat{f}(\lambda_l)\hat{g}(\lambda_l)\chi_l \tag{12}$$

In addition, in this study we introduce a polynomial approximation of the Laplacian matrix based on Chebyshev polynomials to obtain the following effects [3,32]:

- Aggregating the differences between each building and its Kth-order neighbors based on adjacency information;
- Realizing local weight sharing for the convolutional kernels, and;
- Reducing the computational cost for learning.

According to the recursion formula of Chebyshev polynomials, $T_k(X) = 2XT_{k-1}(X) - T_{k-2}(X)$, where $T_0(X) = I_N$ and $T_1(X) = X$, we get the coefficients β_k. The approximation of the Laplacian matrix is designed as [32,34]

$$L = U\begin{bmatrix} \sum_{k=0}^{K} \beta_k \lambda_1^k & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sum_{k=0}^{K} \beta_K \lambda_n^K \end{bmatrix}U^T \tag{13}$$

The following derivation shows a clearer process:

$$L = \beta_1 U \begin{bmatrix} \lambda_1^1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \lambda_n^1 \end{bmatrix} U^T + \ldots + \beta_K U \begin{bmatrix} \lambda_1^K & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \lambda_n^K \end{bmatrix} U^T = \beta_1 \left(U \begin{bmatrix} \lambda_1^1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \lambda_n^1 \end{bmatrix} U^T \right)^1 + \ldots + \beta_K \left(U \begin{bmatrix} \lambda_1^1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \lambda_n^1 \end{bmatrix} U^T \right)^K = \beta_1 L^1 + \ldots + \beta_K L^K = \sum_{j=1}^{K} \beta_j L^j \quad (14)$$

Equation (14) shows that calculating the eigenvectors is not required, which simplifies the computational procedure. To match the requirement that the range of the input eigenvalues is $[-1, 1]$, the following transform is operated on the Laplacian matrix before inputting:

$$\widetilde{L} = \frac{2}{\lambda_{max}} L - I_N \quad (15)$$

where λ_{max} is the maximum of the eigenvalues. In terms of the Laplacian matrix L derived from a graph G, only the coordinate values referring to two adjacent buildings are 1, whereas the others are 0. Therefore, only same-order neighbors share the same weight from the same convolution kernel according to Equation (14), and the property of local weight sharing for kernels is realized, which also enlarges the perception of adjacent regions of the starting building node (Figure 7) for the GCN model with the settable parameter K shown in Equation (14).

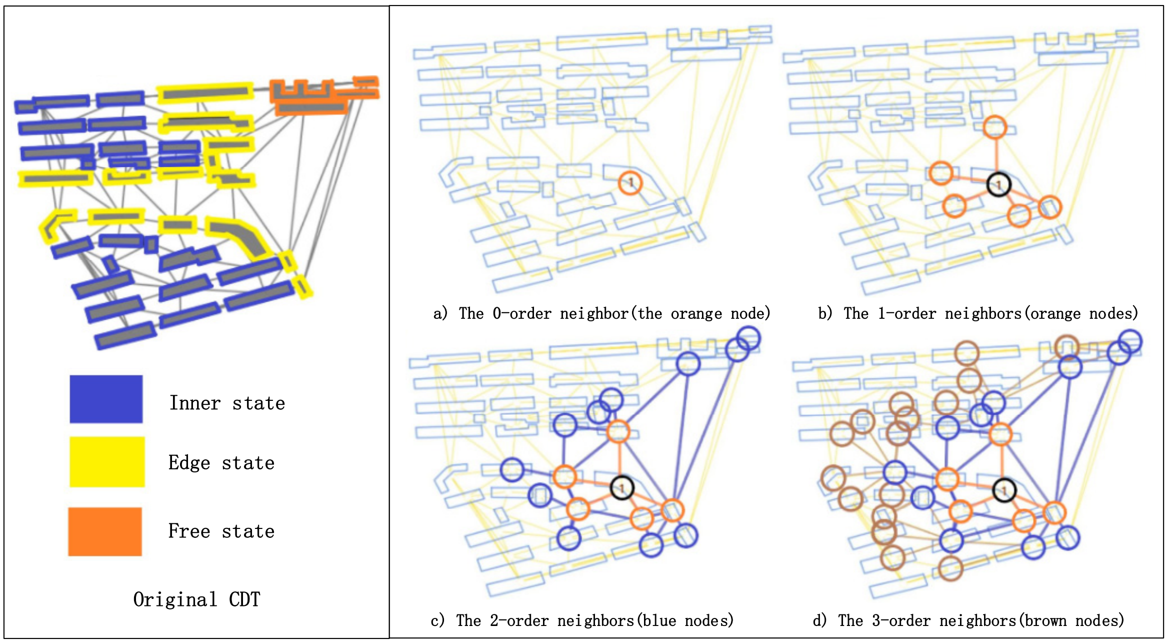
Inner state
Edge state
Free state
Original CDT
a) The 0-order neighbor(the orange node)
b) The 1-order neighbors(orange nodes)
c) The 2-order neighbors(blue nodes)
d) The 3-order neighbors(brown nodes)

Figure 7. Visualization of the perception with different polynomial orders K. (**a**) The 0-order neighbor (the orange node). (**b**) The 1-order neighbors (orange nodes). (**c**) The 2-order neighbors (blue nodes). (**d**) The 3-order neighbors (brown nodes).

Based on the derivation above, the graph convolution formula is given by $f * g = (L\varphi) * g$. After making the low-order polynomial approximation for the Laplacian matrix (Equation (14)), we get the size of the output matrix as (N, Kn), where N is the number of the vertices in a graph and n denotes the length of the descriptive vector of buildings. Therefore, the size of the convolution kernels is (Kn, I), where I refers to the number of convolution kernels. In general, the size of kernels is proportional to the product of K and n, which demonstrates that the computing cost is minimized compared with the classical graph convolution method [36].

Training of the GCN model is based on the gradient descent method to minimize the output loss of the model. According to the chain rule, the gradient expressions used in the back-propagation process [39] are as follows:

$$\left(\frac{bloss}{bw_{i.j}^{(k)}}\right)^{(l)} = x_i^{(k)}\delta_j^{(l+1)}, \qquad i \in \left[0, n_f - 1\right],\ j \in [0, n_k - 1],\ k \in [0, K-1] \tag{16}$$

$$\delta_i^{(k,l)} = \sum_{j=0}^{n_k-1} w_{i,j}^{(k)}\delta_j^{(l+1)} \tag{17}$$

where n_f denotes the length of the input vectors and n_k refers to the number of convolution kernels.

In this study, the GCN model consists of two convolutional layers. The graph convolution operation is shown in Figure 8.

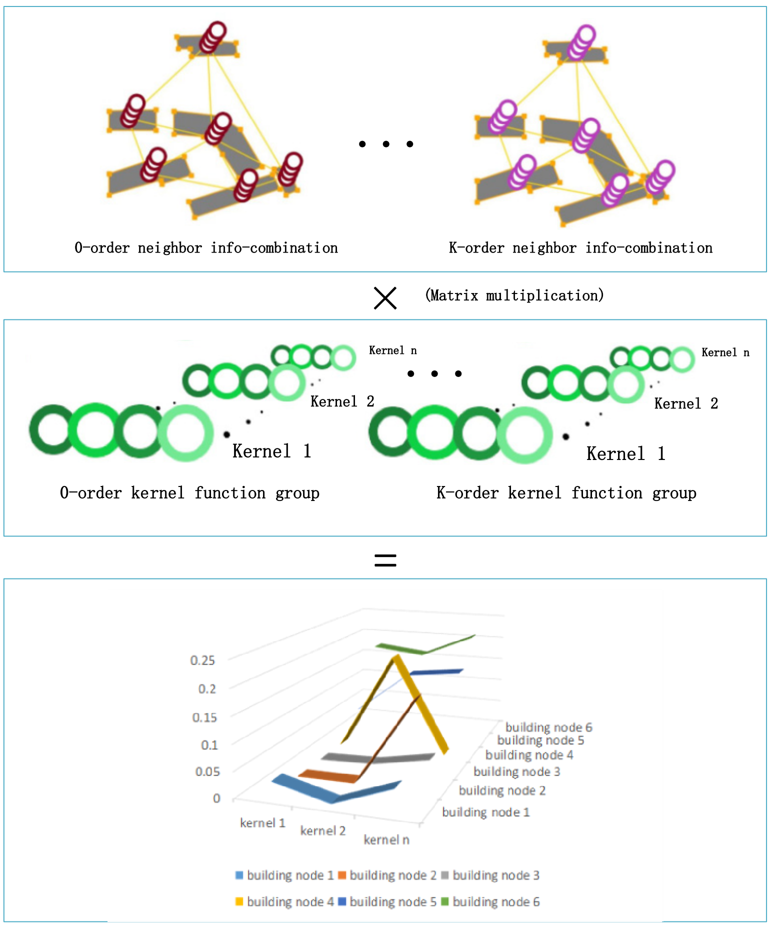
0-order neighbor info-combination
K-order neighbor info-combination
(Matrix multiplication)
Kernel n
Kernel 2
Kernel 1
0-order kernel function group
Kernel n
Kernel 2
Kernel 1
K-order kernel function group
0.25
0.2
0.15
0.1
0.05
0
kernel 1
kernel 2
kernel n
building node 6
building node 5
building node 4
building node 3
building node 2
building node 1
building node 1 building node 2 building node 3
building node 4 building node 5 building node 6

Figure 8. Visualization of the computing process for the graph convolution operation.

There are three kinds of building states, thus there are three convolution kernels in the second layer, and Softmax is the selected as the activation function. Each component of the output vector denotes the probability of the related class.

3.2. Building Group Partition Algorithm

As was stated in Section 3.1, the building nodes in a graph have been classified into three types: edge state nodes, inner state nodes and free state nodes. Through the distribution of buildings is extremely random, after the classification of building states, finite kinds of situations are revealed as follows:

- Situations of inner state: There is only one existing situation of inner state nodes in a building block. As shown in Figure 9b, the inner state nodes are surrounded by the edge state nodes, and form the building block with the latter. Algorithm 2 is used for obtaining the building block containing inner state nodes.
- Situations of edge state nodes: As shown in Figure 9b,e there are two possible situations of edge state nodes: 1. forming the building block with inner state nodes and 2. forming the building block only consisting of edge state nodes. Algorithm 2 is adapted to the first situation, while Algorithm 3 is applied to the second situation.
- Situations of free state nodes: There is only one possible situation of free state nodes because of its independence compared to the other two kinds of building nodes, as shown in Figure 9c. Algorithm 4 is used for constructing the building group consisting only of free state nodes.
- Building nodes in the same state, but not in the same building block: It is possible that, though the nodes are identified as nodes in the same state, they belong to different building blocks (Figure 9d). Moreover, a fine-grained partition is needed because of the differences in the aspects of size, outline and orientation because the group partition step mainly focuses on the spatial distribution. This problem will be solved in the following step.

Algorithm 1 BFS (G, V_s, v, S_{n1})

```
1: Initialize: $S_t$ (an empty stack for the BFS algorithm).
2: push $v$ to $S_t$
3: while not empty ($S_t$) do
4:     $v \leftarrow S_t.pop()$
5:     $l_v$ = [neighbors ($v$)]
6:     for each $v_j \in l_v$ do
7:         if $v_j$'s state is $S_n$ then
8:             append $v_j$ to $V_s$
9:             push $v_j$ to $S_t$
10: Return: $V_s$
```

Algorithm 2 Inner state node searching process for group pattern reconstruction

```
1: Input: Building graph $G = (V, E)$ (where $V[i]$ stores the state of building i recognized by the GCN
(Section 3.1.3)); number of building nodes = $N$
2: Initialize: $V_{Inner}$ (an empty list to store the building groups of inner state), $N_{Inner} = 0$
3: for i = 0 to $N - 1$ do
4:     if $V[i]$ is *Inner* state and have not been appended to $V_{Inner}$ then
5:         $N_{Inner}++$, append(V[i]) to $V_{Inner}[N_{Inner}]$
6:         *BFS* ($G$, $V_{Inner}[N_{Inner}]$, V[i], *Inner* state) ◁*BFS*(Algorithm 1) for node traversing
7: for $i = 0$ to $N - 1$ do
8:     if $V[i]$ is *edge* state and have not been appended to $V_{Inner}$ then
9:         for $j = 0$ to $N_{Inner} - 1$ do
10:            for $k = 0$ to $V_{Inner}[j].length - 1$ do
11:                if V[i] $\in$ neighbors $(V_{Inner}[j][k])$ and $V_{Inner}[j][k]$ is *Inner* state then
12:                    append(V[i]) to $V_{Inner}[j][k]$
13: Return: $V_{Inner}$
```

Algorithm 3 Edge state node searching process for group pattern reconstruction

```
1: Input: Building graph $G = (V, E)$ (where $V[i]$ stores the state of building i recognized by the GCN
(Section 3.1.3)); number of building nodes = $N$
2: Initialize: $V_{edge}$ (an empty list to store building groups of edge state), $N_{edge} = 0$
3: for i = 0 to $N - 1$ do
4:     if $V[i]$ is $edge$ state and have not been appended to $V_{Inner}$ or $V_{edge}$ then
5:         $N_{edge}++$, append(V[i]) to $V_{edge}[N_{edge}]$
6:         $BFS$ (G, $V_{edge}[N_{edge}]$, $V[i]$, $Edge$ state) ◃$BFS$(Algorithm 1) for node traversing
7: Return: $V_{edge}$
```

Algorithm 4 Free state node searching process for group pattern reconstruction

```
1: Input: Building graph $G = (V, E)$ (where $V[i]$ stores the state of building i recognized by the GCN
(Section 3.1.3)); number of building nodes = $N$
2: Initialize: $V_{free}$ (an empty list to store the building groups of free state), $N_{free} = 0$
3: for i = 0 to $N - 1$ do
4:     if V[i] is $free$ state and have not been appended to $V_{free}$ then
5:         $N_{free}++$, append(V[i]) to $V_{free}[N_{free}]$
6:         $BFS$ (G, $V_{free}[N_{free}]$, V[i], $free$ state) ◃$BFS$(Algorithm 1) for node traversing
7: Return: $V_{free}$
```

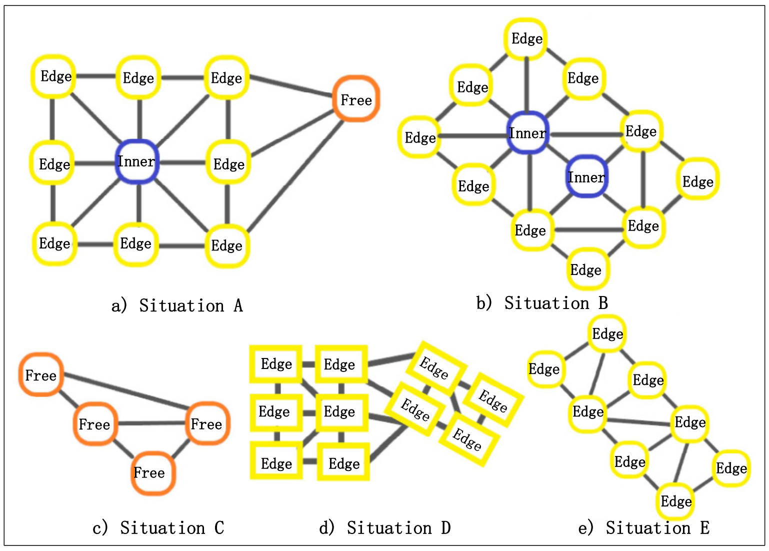

Figure 9. Clustering samples of different state building nodes. (**a**) Situation A. (**b**) Situation B. (**c**) Situation C. (**d**) Situation D. (**e**) Situation E.

3.3. Fine-Grained Partition for Building Blocks

As was stated in Section 3.2, the building group partition step divides building nodes into building blocks according to the spatial distribution. In this study, we use a fine-grained partition method to extract building patterns based on the similarity of the buildings.

We employ indices of the standard deviation (SD) of these building distances [4], the area difference, the orientation difference, the compactness difference and the similarity difference [3], to construct the discriminant vector (Figure 10). The five indices quantify the difference in the related building nodes. In this study, the RF model [4] is utilized to judge whether the buildings are in the same pattern. Algorithm 5 is used for building node clustering to accomplish the fine-grained partition.

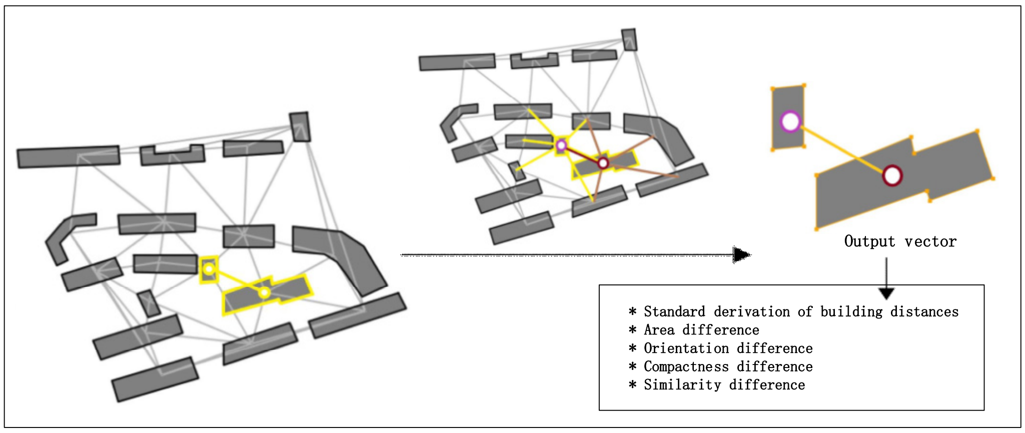

Figure 10. Visualization of the descriptive indices of the difference between building node pairs.

Algorithm 5 Building node clustering

```
1: Input: Building graph $G = (V, E)$ (where $V_i \in V$ means the building nodes from the same building division (Section 3.2)); $N$ means the number of buildings
2: Initialize: $S_l$ (an empty list to store building pattern groups), $M$ (a list initialized by False, storing the state if $V_i$ have been checked), $S_t$ (an empty stack for the BFS algorithm), RF (a function using the RF model to judge whether the buildings should be in the same pattern; see Section 3.3), $L = N$, $n = -1$
3: while $L > 0$ do
4:     $L -$, $n$ ++, $T \leftarrow$ None
5:     append $(V_i)$ to $S_l[n]$, push $V_i$ to $S_t$ ◂ where $V_i \in$ V and M[i] == False
6:     while not empty $(S_t)$ do
7:         $V_i \leftarrow S_t.pop()$, M $[i] \leftarrow$ True
8:         $l_v$ = [neighbors $(V_i)$]
9:         sort $(l_v)$ ◂ by the distances to $V_i$ in ascending order
10:        for each $V_j \in l_v$ do
11:            if $T$ == None and $M[i]$ == False then
12:                if RF $(V_i, V_j)$ is True then
13:                    $T \leftarrow V_j$, append $\left(V_j\right)$ to $S_l[n]$, push $V_j$ to $S_t$, $L -$
14:            else if $T! =$ None and M $[i]$ == False then
15:                if RF $(V_i, V_j$, T) is True then
16:                    append $\left(V_j\right)$ to $S_l[n]$, push $V_j$ to $S_t$, $L -$
17: Return: $S_l$
```

3.4. Building Pattern Recognition

The GCNN model used for building pattern recognition consists of graph convolutional layers and a deep neural network. In this part, we first derive the adjacency information from the coordinate data of building contours through the CDT method. The descriptive vectors are constructed by the shifting degree of the adjacency weight (Section 3.1.2.1) of the key points from the contours. The input matrix is constructed from the descriptive vectors of the key points in a building pattern, and the adjacency matrix recording the adjacency information so that the model only focuses on the topological relationships among the buildings.

The connection of the GCN to the deep neural network is shown as Figure 11. The size of the output matrix of the graph convolutional layer is (N, Kn) (Section 3.1.3), where N is the number of the vertices in a graph and n denotes the length of the descriptive vector of the nodes. The representation vector of a graph G_i is derived by using [40]

$$h_g = \frac{1}{N_{vertex}} \sum_{i=1}^{N_{vertex}} h_v \tag{18}$$

where h_v denotes the output of the graph convolutional layer and h_g is the input of the fully connected layer. The training method is the same as mentioned in Section 3.1.3.

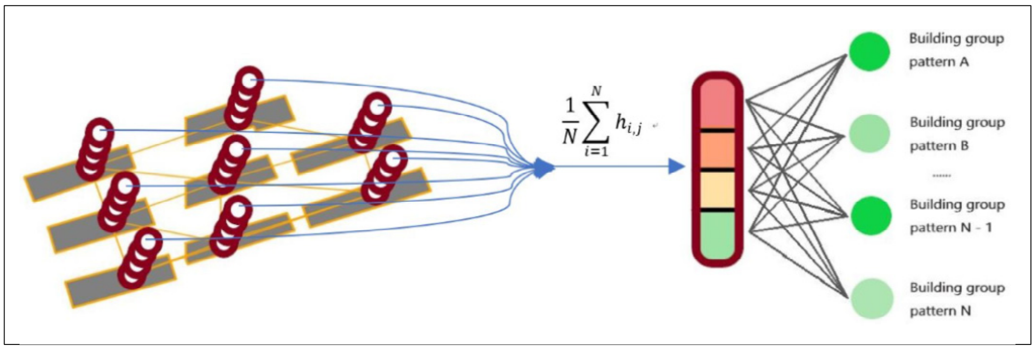

Figure 11. Visualization of the method of connecting the convolutional layer and the fully connected layer.

4. Framework for Building Pattern Recognition

In general, based on the methods described in Section 3, to achieve building pattern recognition, the algorithm framework proposed in this study consists of the following five parts:

- Graph construction for buildings. Each building node has its own unique identification number. As shown in Figure 12a,b we first derive the adjacency information through constructing a constrained Delaunay triangulation (CDT) [4,41,42] for all the buildings.
- Building node state identification. Based on the vector data of building contours, each building entity is described by indices including its area, perimeter, orientation, compactness and shifting degree of adjacency weight (Section 3.1.2.1), and then a descriptive vector is constructed. The descriptive vectors are the input to the GCN model in the subsequent step. To make rules for building group partition, three states (edge state, inner state and free state) (Section 3.1.1) are defined to describe the spatial state of the buildings. The related dataset of building node state labeling is constructed, and the GCN model with semi-supervised learning is trained to enhance the ability of generalization. The GCN model is used to identify the building node state, which is indispensable for the building group partition algorithm (see Section 3.2). The partition process and the building samples identified as different states are shown in Figure 12c.
- Building group partitioning. The building group partition algorithm based on the identification results of the building state is run. The outputs of the algorithm are the building blocks from the partitioned building groups (Figure 12d).
- Building node clustering. A breadth-first search (BFS) is used to traverse building nodes in a graph, and the graph of each building block is constructed by CDT. A random forest (RF) algorithm is introduced to judge whether two or three building entities (Section 3.3) can be categorized into the same building pattern. The objective of this step is to extract all the separate building patterns in a building block (Figure 12e).
- Building pattern recognition. In this final step, a GCNN model (Section 3.4) is used to recognize the building patterns, as shown in Figure 12f. The model is trained with supervised learning with the building node pattern datasets.

To make a more intuitive description, the specific process of the framework is summarized in Algorithm 6, and the whole workflow is shown in Figure 12.

Algorithm 6 Framework for building pattern recognition

```
1: Require:
2:     $X$: denotes the data of a building block selected in advance.
3:     $X_j$: denotes primary data of a building object with ID $j$, including the coordinate data of
4:         the building contours.
5: Step 1: Construct a graph $G$ for $X$ using CDT.
6: Step 2: Calculate the values of the variables mentioned in Section 3.1.2 to construct a
7: new descriptive vector for each building object $X_j$, on the basis of the adjacent relations
8: derived from the graph $G$.
9: Step 3: Classify each building $X_j$ into the state $S_j$ by the GCN model (see Section 3.1.3).
10: Step 4: Accomplish building graph partition. Functions $f_{a2}$, $f_{a3}$ and
11: $f_{a4}$ stand for Algorithms 2–4 respectively.
12:     $V_{Inner} = f_{a2}(G)$
13:     $V_{edge} = f_{a3}(G)$
14:     $V_{free} = f_{a4}(G)$
15: Where $V_{Inner}[i]$ denotes the $i^{th}$ building group and $V_{Inner}[i]$ = [ $X_0 \ldots X_{k-1}$]. $k$ stands for
16: the number of the buildings of the $i^{th}$ building group. Data structures of $V_{edge}$ and
17: $V_{free}$ are the same as $V_{Inner}$.
18: Step 5: Utilize the RF model to accomplish the fine-grained partition (Section 3.3) for
19: the building groups from $V_{Inner}$, $V_{edge}$ and $V_{free}$.
20: Initialize:
21:     $S_l$: an empty list.
22:     $V$: a list consists of all the building groups from $V_{Inner}$, $V_{edge}$ and $V_{free}$.
22: for $i$ = 0 to $V.length - 1$ do
23:     $G' = V[i]$
24:     $S' = f_{a5}(G')$ ◃ $f_{a5}$ stands for the function of Algorithm 5
25:     for j = 0 to $S'.length - 1$ do
26:         append $S'[j]$ to $S_l$ ◃ $S'[j]$ denotes a building group
27:     end for
28: end for
29: Step 6: Classify the building group into the pattern with the GCNN model.
30: for $i$ = 0 to $S_l.length - 1$ do
31:     $p' = GCNN(S_l[i])$ ◃ $p'$ means the classifying result
32: end for
```

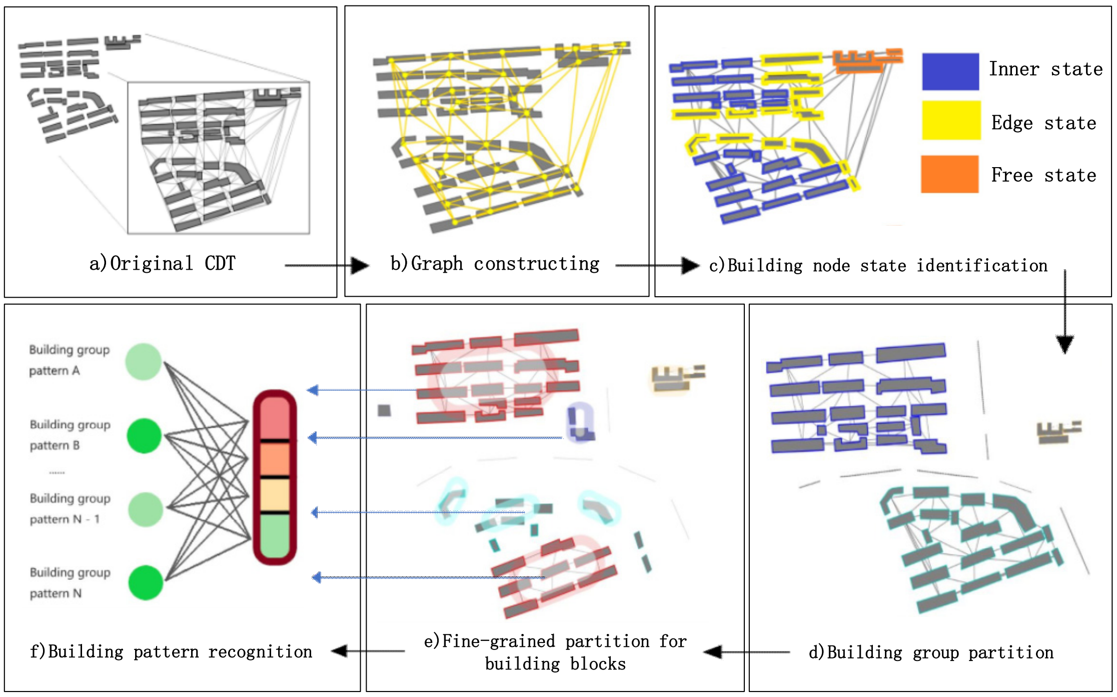
a)Original CDT
b)Graph constructing
c)Building node state identification
Inner state
Edge state
Free state
d)Building group partition
e)Fine-grained partition for building blocks
f)Building pattern recognition
Building group pattern A
Building group pattern B
Building group pattern N - 1
Building group pattern N

Figure 12. Description of building group pattern abstraction and detection algorithm framework. (**a**) Original CDT. (**b**) Graph constructing. (**c**) Building node state identification. (**d**) Building group partition. (**e**) Fine-grained partition for building blocks. (**f**) Building pattern recognition.

5. Experiments and Results

5.1. Building Node State Recognition

In the task of building state identification (Section 3.1.3), the output of the GCN model is a probability vector $\{P_i\}_{i=0}^{M-1}$, where M denotes the number of possible classes. If P_i is the maximum, then i refers to the class in which the object belongs. As was stated in Section 4, the graph generated by CDT is the input of the GCN model. The model structure, shown in Figure 13, consists of two convolutional layers with 64 kernels and 3 kernels, respectively. The Softmax function is selected as the activation function of the last layer, and its output is a probability vector. During the training process, we use the Adam optimization algorithm as the optimizer. The regularization weight is set as 5×10^{-4}. To enhance the generalization ability, the model is trained with semi-supervised learning [15], and the dropout probability is set as 0.5. Meanwhile, the model is trained by one graph per step.

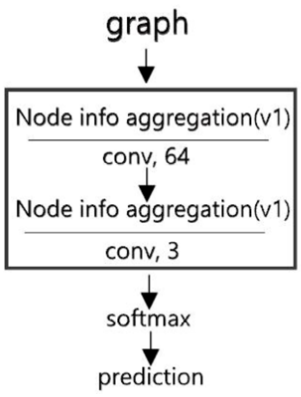

Figure 13. Structure of the graph convolutional network (GCN) model for building node state identification. Node info aggregation(v1) refers to using the low-order polynomial approximation method (Section 3.4).

As shown in Figure 14, after training on the Beijing Xicheng District dataset for 40 epochs, the training accuracy achieved 86.05% with the semi-supervised learning algorithm. The testing accuracy on the Xi'an dataset achieved 92.71%, and the testing loss was 0.362.

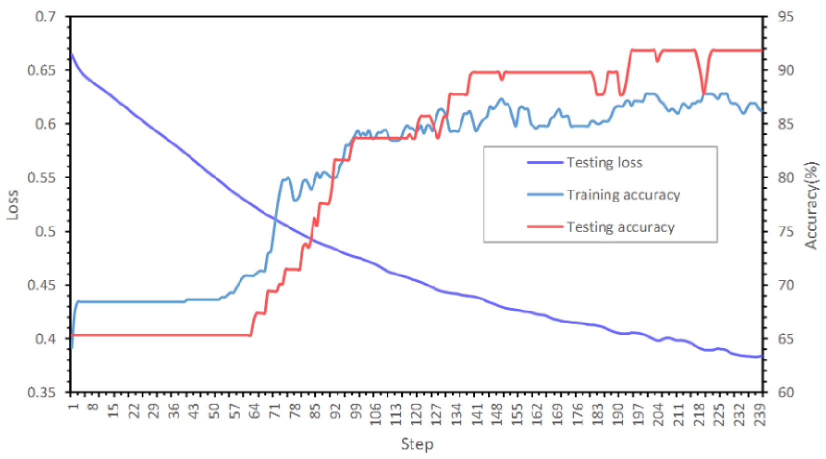

Figure 14. Training and testing results from the GCN model for building node state identification.

The confusion matrix is shown in Table 2, and the kappa coefficient is 0.832. Given that the training set and the testing are based on data from two typical cities in China, the accuracies and the two curves indicate the good generalization ability of the GCN model. Figure 15b shows the partial results of the trained model on the testing data. The buildings with blue contours are identified as edge state buildings.

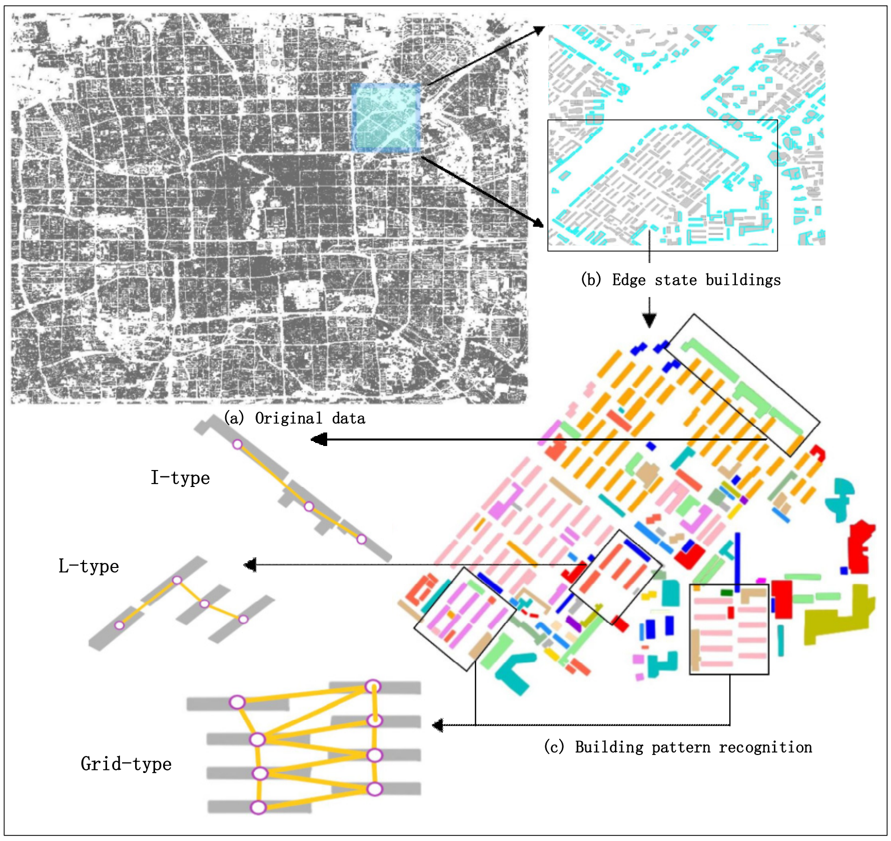

Figure 15. Partial results from the proposed algorithm framework.

Table 2. Confusion matrix of the building node state identification experiment with the dataset of the core areas of Xi'an.

Number of Examples = 950	Actual Inner State	Actual Edge State	Actual Free State
Predicted inner state	668	24	1
Predicted edge state	37	184	2
Predicted free state	0	0	34

Figure 16 shows the training accuracies and losses with different polynomial orders K. The results indicate that the performance is poor when $K = 1$, but performance begins to improve from $K = 2$ to $K = 3$. When $K = 4$, the accuracy decreases. This confirms that a larger perception region helps improve the accuracy. In addition, the findings indicate that the best performance emerges when $K = 3$, with higher values of K having an adverse impact.

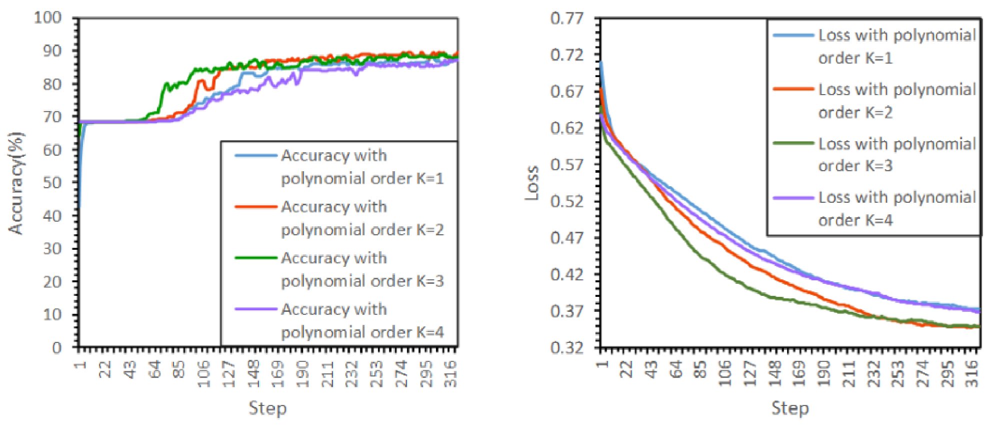

Figure 16. Test results from the GCN model for building node state identification with different polynomial orders K.

5.2. Fine-Grained Partition

The RF model is employed to judge whether the buildings belong to the same building pattern for the fine-grained partition task (Section 3.3). The Beijing Xicheng District dataset is used for model training, while the Xi'an dataset is used for testing. The input vectors are prepared by following the steps described in Section 3.3, and the RF model is trained with supervised learning. In addition, a support vector machine (SVM) model is trained under the same conditions for comparison [3]. The experimental results, given in Table 3, indicate the RF method achieves better performance. The experiment confirms that the effect of the proposed method is satisfactory, and the generalization ability of the RF model is strong enough to be applied to other datasets.

Table 3. Comparative results of the two methods for the building node clustering task.

Method	Training Accuracy (Beijing Xicheng District)	Testing Accuracy (Xi'an)
SVM	98.30%	84.35%
RF	99.06%	96.77%

5.3. Building Group Pattern Recognition and Comparative Analysis

The structure of the GCNN model used for the building pattern recognition task is shown in Figure 17. In this experiment, the Beijing Xicheng District dataset is used for model training and the Xi'an dataset is used for testing. Table 4 shows the results: the accuracies of training and testing are 98.20% and 89.83%, respectively. In addition, the confusion matrix of the recognition result on the testing set (Xi'an dataset) is shown in Table 5. The kappa coefficient is 0.847. Two samples of recognition results on the testing set are shown in Figures 15c and 18.

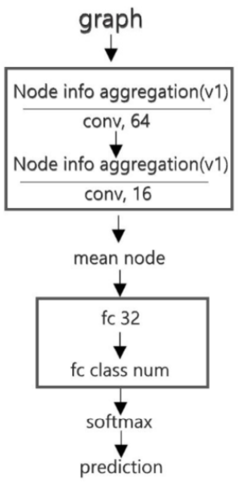

Figure 17. Structure of the graph convolutional neural network (GCNN) model. Node info aggregation(v1) refers to using the low-order polynomial approximation method (Section 3.1.3).

Table 4. Comparative results of the three methods for the building pattern recognition task.

Method	Training Accuracy (Beijing Xicheng District)	Testing Accuracy (Xi'an)
SVM	99.68%	77.18%
RF	99.45%	81.78%
GCNN	98.20%	89.83%

Table 5. Confusion matrix of the building pattern recognition experiment with the dataset of the core areas of Xi'an.

Number of Examples = 354	Actual I-Shape	Actual L-Shape	Actual Grid-Like
Predicted I-shape	118	9	11
Predicted L-shape	6	109	7
Predicted Grid-like	1	2	91

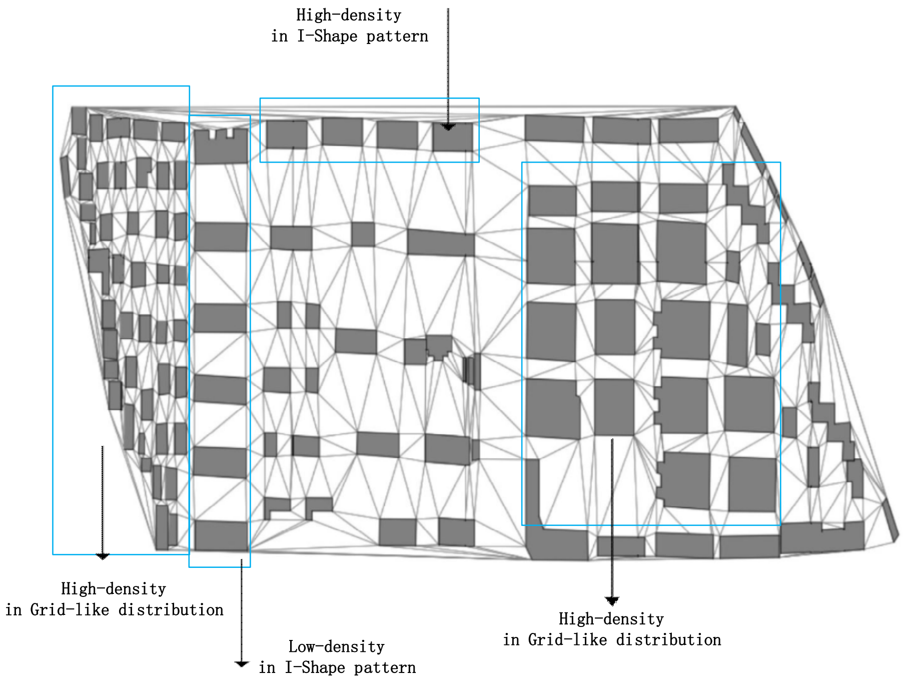

Figure 18. Part of the pattern recognition results with different densities.

The following comparative experiment was done for better testing the advantages and disadvantages of the proposed method. Indices of mean distance [4], standard deviation of building distances [4], black-and-white ratio [43] and area ratio of the building to the smallest bounding rectangle of the group [4] are employed to construct the descriptive vectors for the building patterns. The SVM model and the RF model were utilized for comparison, and the descriptive vectors of the building patterns extracted from the datasets were used as the input of the models. The comparative results are listed in Table 4. Both the SVM method and the RF method have high training accuracy, but their testing accuracy is not good. The results can be explained by the fact that the descriptive vector input to the SVM model and the RF model leads to the sensitivity to spatial distribution of the buildings in various geographical regions, while the random distribution of the buildings in the datasets leads to the density being extremely not stationary. As shown in Figure 18, the building patterns that are the same class but have different density of buildings cause difficulties for the models in learning generalization features and identification rules. By contrast, the GCNN model in this study only focuses on the adjacency relationship.

Therefore, the proposed method is not spatial distribution sensitive, especially when the training data and testing data are not from the same dataset.

As in the experiment above, five indices (standard deviation of building distances, similarity difference, area difference, orientation difference and compactness difference among the building and its neighbors) [3] are utilized to construct the descriptive vector for one building node. The vectors are used as the input to the SVM model and the RF model. The two models are trained by using the same datasets as for the GCN model (Section 5.1). Table 6 lists the comparative results, which indicate that the accuracy of the GCN method is the best on the testing dataset, while the generalization ability of the other two methods is relatively poor.

Table 6. Comparative results of the three methods for the building node state identification task.

Method	Training Accuracy (Beijing Xicheng District)	Testing Accuracy (Xi'an)
SVM	94.52%	81.35%
RF	96.84%	89.39%
GCNN	86.05%	92.71%

5.4. Parameter Descriptive Ability Analysis

In this experiment, we examined the descriptive ability of each descriptive variable. We used a single variable to construct the input vectors and repeated the experimental steps of building node state identification. The results shown in Figure 19 indicate that the testing accuracy (90.8%) is close to the accuracy (92.71%) of the original experiment (Section 5.1) when the input vectors are constructed only by the shifting degree of adjacency weight (Section 3.1.2.1). Therefore, it can be inferred that, for the task of building group partition that is related to human visual perception, the descriptive ability of the shifting degree of adjacency weight is effective. In addition, the findings also confirm that the concept of using the shifting degree of adjacency weight is reasonable in machine-learning tasks based on topological graphs when relating to visual perception.

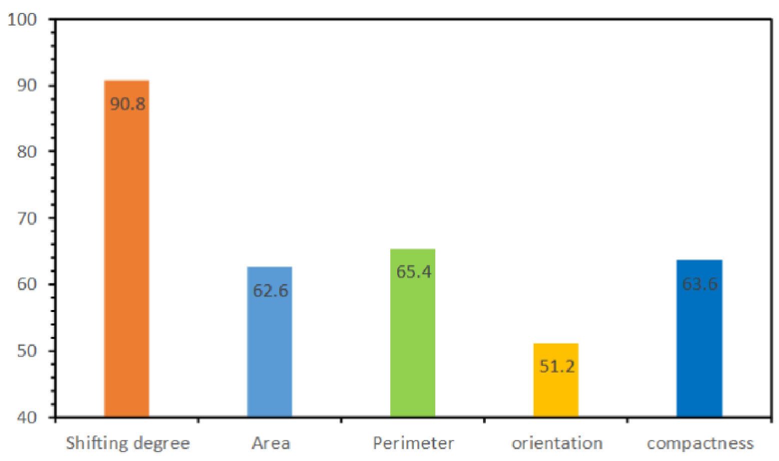

Figure 19. Experimental results for the descriptive abilities of different indices.

5.5. Model Structure Exploration Results

Different strategies of model structure have different applicability in the tasks of building state identification and building pattern recognition. This experiment is aimed to explore the difference of the performances with different model structures. Figure 20 shows the model structures and the related experimental results using the Xi'an dataset for testing.

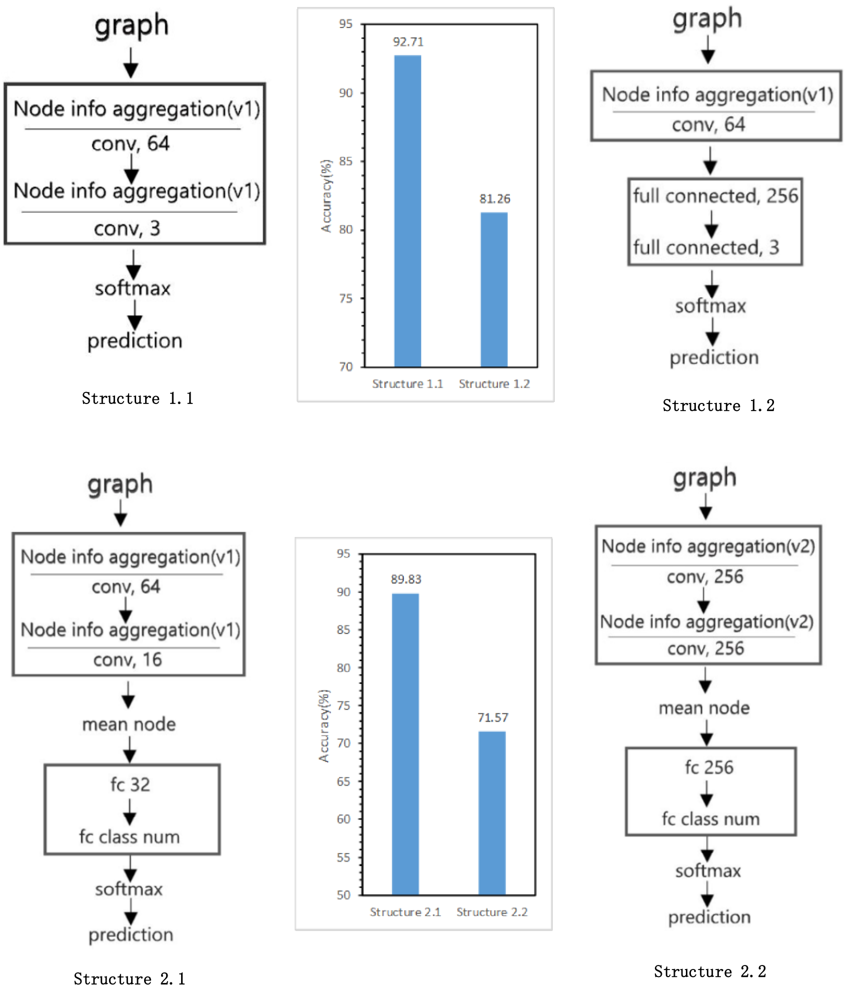

Figure 20. Experimental results for network structure exploration.

Figure 20 (Structures 1.1 and 1.2) show that the classification ability of the fully connected layer is not good compared with the graph convolutional layer in the building state identification task, and the total precisions are 81.26% and 92.71%, respectively. Figure 20 (Structures 2.1 and 2.2) show that, in the task of building pattern recognition, using the convolution operation described in Section 3.1.3 offers better performance than the classical graph convolution method [36], as the accuracies on the testing set are 89.83% and 71.57%, respectively, while the latter requires a greater amount of computation.

6. Discussion

6.1. Spatial Adaptive Algorithm Framework Using GCNs

The GCN model and the GCNN model is not spatial distribution sensitive in the building pattern recognition task, because they only focus on the adjacency information and the characteristic differences between the building and its neighbors. By comparison, the RF-based method [4] is most likely limited because the variables input measure the indices in Euclidean space directly which lead to a bad performance, since some spatial features, such as the sparsity of the building blocks, sizes and the geometries of the buildings are very flexible and various. The experiments have demonstrated that our method for building pattern recognition outperforms the existing related methods, especially from the aspects of generalization ability and testing accuracy.

In addition, by combining the GCN model and the proposed algorithms, this bottom-up method can perceive the relationships among the building and its neighbors in the process of building a group partition, without the requirement of ancillary data (e.g., road networks and rivers).

6.2. Remaining Issues

As seen in Section 5, though the GCN model and the GCNN model are better in terms of generalization ability and testing results, their training accuracies are not satisfactory. One of the reasons for this is the existence of ambiguous situations during the annotation process.

A sample is shown in Figure 21. Intuitively, buildings (a) and (b) should be in the edge state (Section 3.1.1), given their position in the building block. However, given the adjacency information and the small shifting degree of adjacency weight (Section 3.1.2.1), these buildings can reasonable be identified as inner state buildings. Such ambiguous situations lead to difficulty in making a precise graph dataset, destabilizing training and decreasing accuracy.

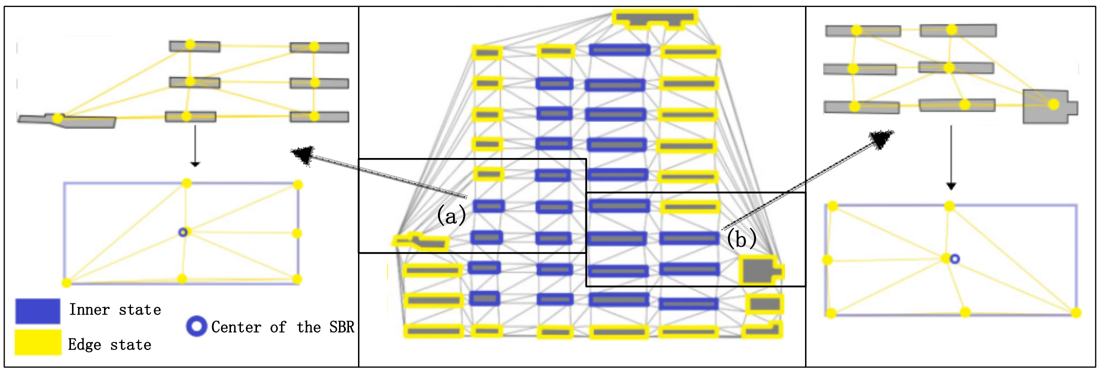

Figure 21. Recognition results from the GCN model for building state identification.

In addition, CDT is constructed for the coordinate data of the building contours as the input to the GCNN model (Section 3.4). However, another method of constructing CDT based on the center points of the building nodes [3] is not employed in this study, because the performance is often poor when the GCNN model is used in small graphs.

Therefore, an obvious limitation of the proposed method is that the topological structures of the same building patterns are not always stationary because of the various shapes of the buildings, and this leads to difficulty for the learning of the model.

7. Conclusions and Future Works

7.1. Conclusions

In this research, an algorithm framework for building pattern extraction and recognition combining the graph convolution operation, the RF model, a neural network and spatial adaptive algorithms, has been proposed. Besides, the multi-stage design of the framework is to achieve building pattern extraction which is associated with the multi-object detection task on topological data. The shifting degree of adjacency weight proposed in this research is utilized in order to exploit the distribution features of the building nodes and spatial adjacent relations. Experiments confirm the effectiveness of the descriptive vector constructed by the shifting degree of adjacency weight and other variables (e.g., the area, perimeter, orientation and compactness). Additionally, training and testing results indicate good generalization ability of the GCN model, since the training set and testing set are derived from the two various regions, Beijing's Xicheng District and the core areas of the city of Xi'an, which shows that the proposed method is not spatial distribution sensitive. Another superiority is that this framework enables the building group partition task to be performed without any ancillary data. In addition, our study confirms the feasibility of using the graph convolution method to address the problem of building pattern recognition through a sample experimental study.

7.2. Future Works

We explored the applicability of different model structures and derived an ideal effect with reasonable computation cost. In the future works, the improvement of the aggregation operation (Section 3.1.3) is one of the important options for increasing the accuracy, since we simply calculate the gradients of the vectors in the research. More effort will be put into devising better descriptive methods for buildings and exploring more powerful models. In addition, as is stated in Section 6.2, the performance degradation of the GCNN model in small graphs is worthy of research in the future. Last but not least, solving the problem of ambiguous situations during the annotation process is essential, otherwise the performance of the models will be hard to be improved for the lack of precise graph datasets.

Author Contributions: Conceptualization, W.B., and M.G.; Methodology, W.B., and M.G.; Project administration, M.G.; Software, W.B.; Supervision, M.G.; Visualization, W.B.; Writing—original draft, W.B.; Writing—review and editing, M.G. and Y.H.; Funding acquisition, M.G.

Funding: This work was supported by the National Natural Science Foundation of China (Nos. 41701446, 41971356) and the Hubei Province Natural Science Foundation of China (No. 2017CFB277).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Du, S.; Luo, L.; Cao, K.; Shu, M. Extracting building patterns with multilevel graph partition and building grouping. *ISPRS J. Photogramm. Remote Sens.* **2016**, *122*, 81–96. [CrossRef]
2. Du, S.; Shu, M.; Feng, C. Representation and discovery of building patterns: A three-level relational approach. *Int. J. Geogr. Inf. Sci.* **2016**, *30*, 1161–1186. [CrossRef]
3. Yan, X.; Ai, T.; Yang, M.; Yin, H. A graph convolutional neural network for classification of building patterns using spatial vector data. *ISPRS J. Photogramm. Remote Sens.* **2019**, *150*, 259–273. [CrossRef]
4. He, X.; Zhang, X.; Xin, Q. Recognition of building group patterns in topographic maps based on graph partitioning and random forest. *ISPRS J. Photogramm. Remote Sens.* **2018**, *136*, 26–40. [CrossRef]
5. Regnauld, N. Contextual Building Typification in Automated Map Generalization. *Algorithmica* **2001**, *30*, 312–333. [CrossRef]

6. Li, Z.; Yan, H.; Ai, T.; Chen, J. Automated building generalization based on urban morphology and Gestalt theory. *Int. J. Geogr. Inf. Sci.* **2004**, *18*, 513–534. [CrossRef]
7. Deng, M.; Tang, J.; Liu, Q.; Wu, F. Recognizing building groups for generalization: A comparative study. *Cartogr. Geogr. Inf. Sci.* **2018**, *45*, 187–204. [CrossRef]
8. Gonzalez-Abraham, C.E.; Radeloff, V.C.; Hammer, R.B.; Hawbaker, T.J.; Stewart, S.I.; Clayton, M.K. Building patterns and landscape fragmentation in northern Wisconsin, USA. *Landsc. Ecol.* **2007**, *22*, 217–230. [CrossRef]
9. Girshick, R. Fast R-CNN. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 13–16 December 2015; pp. 1440–1448.
10. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In *Advances in Neural Information Processing Systems 28*; Cortes, C., Lawrence, N.D., Lee, D.D., Sugiyama, M., Garnett, R., Eds.; Curran Associates, Inc.: Montreal, QC, Canada, 2015; pp. 91–99.
11. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 26 June 2016; pp. 779–788.
12. Girshick, R.; Donahue, J.; Darrell, T.; Malik, J. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Columbus, OH, USA, 24–27 June 2014; pp. 580–587.
13. He, K.; Gkioxari, G.; Dollar, P.; Girshick, R. Mask R-CNN. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017; pp. 2961–2969.
14. Long, J.; Shelhamer, E.; Darrell, T. Fully Convolutional Networks for Semantic Segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 7–12 June 2015; pp. 91–99.
15. Kipf, T.N.; Welling, M. Semi-Supervised Classification with Graph Convolutional Networks. *arXiv* **2016**, arXiv:1609.02907.
16. Fout, A.; Byrd, J.; Shariat, B.; Ben-Hur, A. Protein Interface Prediction using Graph Convolutional Networks. In *Advances in Neural Information Processing Systems 30*; Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R., Eds.; Curran Associates, Inc.: Long Beach, CA, USA, 2017; pp. 6530–6539.
17. Parisot, S.; Ktena, S.I.; Ferrante, E.; Lee, M.; Guerrero, R.; Glocker, B.; Rueckert, D. Disease prediction using graph convolutional networks: Application to Autism Spectrum Disorder and Alzheimer's disease. *Med. Image Anal.* **2018**, *48*, 117–130. [CrossRef]
18. Ping, X.; Shuxiang, P.; Tiangang, Z.; Yong, L.; Hao, S. Graph Convolutional Network and Convolutional Neural Network Based Method for Predicting lncRNA-Disease Associations. *Cells* **2019**, *8*, 1012.
19. Fei, M.; Fei, G.; Jinping, S.; Huiyu, Z.; Amir, H. Attention Graph Convolution Network for Image Segmentation in Big SAR Imagery Data. *Remote Sens.* **2019**, *11*, 2586.
20. Anselin, L. Local Indicators of Spatial Association—LISA. *Geogr. Anal.* **1995**, *27*, 93–115. [CrossRef]
21. Tobler, W.R. A Computer Movie Simulating Urban Growth in the Detroit Region. *Econ. Geogr.* **1970**, *46*, 234–240. [CrossRef]
22. Zhang, X.; Ai, T.; Stoter, J.; Kraak, M.; Molenaar, M. Building pattern recognition in topographic data: Examples on collinear and curvilinear alignments. *Geoinformatica* **2013**, *17*, 1–33. [CrossRef]
23. Hamilton, W.; Ying, Z.; Leskovec, J. Inductive Representation Learning on Large Graphs. In *Advances in Neural Information Processing Systems 30*; Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R., Eds.; Curran Associates, Inc.: Long Beach, CA, USA, 2017; pp. 1024–1034.
24. Gou, J.; Qiu, W.; Yi, Z.; Xu, Y.; Mao, Q.; Zhan, Y. A Local Mean Representation-based K-Nearest Neighbor Classifier. *ACM Trans. Intell. Syst. Technol.* **2019**, *10*, 21–29. [CrossRef]
25. Gou, J.; Wang, L.; Hou, B.; Lv, J.; Yuan, Y.; Mao, Q. Two-phase probabilistic collaborative representation-based classification. *Expert Syst. Appl.* **2019**, *133*, 9–20. [CrossRef]
26. Gou, J.; Hou, B.; Yuan, Y.; Ou, W.; Zeng, S. A new discriminative collaborative representation-based classification method via l2 regularizations. *Neural Comput. Appl.* **2019**, 1–15. [CrossRef]
27. Chen, Y.N. Multiple Kernel Feature Line Embedding for Hyperspectral Image Classification. *Remote Sens.* **2019**, *11*, 2892. [CrossRef]

28. Perozzi, B.; Al-Rfou, R.; Skiena, S. DeepWalk: Online Learning of Social Representations. In Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY, USA, 24–27 August 2014; pp. 701–710.
29. Grover, A.; Leskovec, J. Node2Vec: Scalable Feature Learning for Networks. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August 2016; pp. 855–864.
30. Tang, J.; Qu, M.; Wang, M.; Zhang, M.; Yan, J.; Mei, Q. LINE: Large-scale Information Network Embedding. In Proceedings of the 24th International Conference on World Wide Web, Florence, Italy, 18–22 May 2015; pp. 1067–1077.
31. Wang, D.; Cui, P.; Zhu, W. Structural Deep Network Embedding. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, San Francisco, CA, USA, 13–17 August 2016; pp. 1225–1234.
32. Defferrard, M.; Bresson, X.; Vandergheynst, P. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. In *Advances in Neural Information Processing Systems 29*; Lee, D.D., Sugiyama, M., Luxburg, U.V., Guyon, I., Garnett, R., Eds.; Curran Associates, Inc.: Barcelona, Spain, 2016; pp. 3844–3852.
33. Babyak, M.A. What you see may not be what you get: A brief, nontechnical introduction to overfitting in regression-type models. *Psychosom. Med.* **2004**, *66*, 411–421.
34. Hammond, D.K.; Vandergheynst, P.; Gribonval, R. Wavelets on graphs via spectral graph theory. *Appl. Comput. Harmon. A* **2011**, *30*, 129–150. [CrossRef]
35. Scarselli, F.; Gori, M.; Tsoi, A.C.; Hagenbuchner, M.; Monfardini, G. The Graph Neural Network Model. *IEEE Trans. Neural Netw.* **2009**, *20*, 61–80. [CrossRef] [PubMed]
36. Duvenaud, D.K.; Maclaurin, D.; Iparraguirre, J.; Bombarell, R.; Hirzel, T.; Aspuru-Guzik, A.; Adams, R.P. Convolutional Networks on Graphs for Learning Molecular Fingerprints. In *Advances in Neural Information Processing Systems 28*; Cortes, C., Lawrence, N.D., Lee, D.D., Sugiyama, M., Garnett, R., Eds.; Curran Associates, Inc.: Montreal, QC, Canada, 2015; pp. 2224–2232.
37. Basaraner, M.; Cetinkaya, S. Performance of shape indices and classification schemes for characterising perceptual shape complexity of building footprints in GIS. *Int. J. Geogr. Inf. Sci.* **2017**, *31*, 1952–1977. [CrossRef]
38. Peura, M.; Iivarinen, J. Efficiency of Simple Shape Descriptors. In *Advances in Visual form Analysis: Proceedings of the 3rd International Workshop on Visual Form*; World Scientific: Capri, Italy, 1997; pp. 443–451.
39. LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-Based Learning Applied to Document Recognition. *Proc. IEEE* **1998**, *86*, 2278–2324. [CrossRef]
40. Gilmer, J.; Schoenholz, S.S.; Riley, P.F.; Vinyals, O.; Dahl, G.E. Neural Message Passing for Quantum Chemistry. In Proceedings of the 34th International Conference on Machine Learning, JMLR.org, Sydney, NSW, Australia, 6–11 August 2017; pp. 1263–1272.
41. Jones, C.B.; Bundy, G.L.; Ware, M.J. Map Generalization with a Triangulated Data Structure. *Am. Cartogr.* **1999**, *22*, 317–331.
42. Touya, G.; Coupé, A.; Jollec, J.L.; Dorie, O.; Fuchs, F. Conflation Optimized by Least Squares to Maintain Geographic Shapes. *ISPRS Int. J. Geo-Inf.* **2013**, *2*, 621–644. [CrossRef]
43. Zhang, X.; Ai, T.; Stoter, J. Characterization and Detection of Building Patterns in Cartographic Data: Two Algorithms. In *Advances in Spatial Data Handling and GIS*; Springer: Berlin/Heidelberg, Germany, 2012; pp. 93–107.