Urban Architectural Landscape Recognition Based on Deep Convolutional Neural Networks and LiDAR

Xiaoyong Zhang[1], Shan Mi[2*], Xiuhuan Feng[3] and Weiying Ding[4]

[1]Modern education technology center, Tangshan Normal University, Tangshan 063000, Hebei, China, zxy@tstc.edu.cn
[2]Academy of Arts, North China University of Science and Technology, Tangshan 063210, Hebei, China, tsmishan@163.com
[3]Department of History, Culture, and Law, Tangshan Normal University, Tangshan 063000, Hebei, China, zsxfengxiuhuan@126.com
[4]Department of Computer Science, Tangshan Normal University, Tangshan 063000, Hebei, China, weiyingding@163.com

[*]Corresponding author: Shan Mi, tsmishan@163.com

Abstract: For today's rapidly developing cities, identifying and managing urban architectural landscapes is an important task in urban planning, environmental monitoring, and disaster prevention. Traditional building identification methods mainly rely on manual surveying and on-site investigation, which is not only time-consuming and labor-intensive but also easily affected by human and environmental factors, resulting in inaccurate and incomplete identification results. This article introduced an urban architectural landscape recognition method that combined Deep Convolutional Neural Network (DCNN) and Light Detection and Ranging (LiDAR) technology. During the research process, by fusing multi-level features from images and LiDAR data, high-precision and efficient feature extraction was achieved, improving recognition precision and model generalization ability. Afterward, the ResNet (Residual Network) 50 model was combined with the extracted advanced features to form the model in this article. The model performance was improved through He parameter initialization, classification cross-entropy loss function, and Adam optimizer. The research results indicated that the model in this article could achieve good convergence in about 50 rounds during the training process, and the accuracy in testing also reached about 93%, which was a significant improvement compared to traditional model recognition. This research not only provides new ideas for the recognition of urban architectural landscapes and promotes their development but also offers reference methods and data support for studies in related fields.

Keywords: Deep Learning; Convolutional Neural Networks; Image Recognition; Multi-source Data Fusion; Light Detection and Ranging; Urban Architectural Landscape Recognition
DOI: https://doi.org/10.14733/cadaps.2025.S3.259-271

1. INTRODUCTION

With the continuous growth of the population and the rapid development of urban construction, the scale of urban buildings is constantly expanding, and the issues of recognition and management of various buildings within the city are exposed to the administrators. Important links of urban construction such as urban planning, infrastructure construction, environmental monitoring and emergency management all rely on relevant data of buildings within the city. Therefore, the recognition of urban architectural landscapes is indispensable. Traditional methods rely on manual feature extraction and traditional machine learning algorithms, which are inefficient and perform poorly in complex environments. Manual feature extraction requires a lot of professional knowledge and human resources and is difficult to handle large-scale data, while traditional machine learning has limitations in high-dimensional data and multi-source data fusion [1-2]. The development of LiDAR technology has made it possible to obtain high-precision 3D spatial information for image recognition [3], but LiDAR data alone cannot fully capture architectural diversity and complexity [4]. The color, texture and edge features provided by image data compensate for LiDAR's lack of 2D visual information. Combining LiDAR and image data can fully and accurately describe the urban architectural landscape. Over recent years, there have been many researchers focusing on deep learning, especially DCNN, because it has brought a great change in image recognition, providing new tools for automatic extraction and fusion of multi-source data [5]. Compared with traditional manual feature extraction, DCNN has obvious advantages in terms of automation and robustness of feature extraction and performs well when dealing with large-scale data and complex pattern recognition tasks. DCNN automatically extracts multi-level features in image and LiDAR data through multi-layer convolution operations, improving the efficiency and accuracy of feature extraction, which brings the possibility of multi-source data fusion. Multi-source data fusion utilizes the advantages of various types of data to make up for the deficiencies of a single data source, achieving more precise and reliable identification [6]. Therefore, combining DCNN and LiDAR techniques for urban architectural landscape recognition is a research direction that most researchers have agreed upon.

In this article, an efficient architectural landscape recognition method combining DCNN and LiDAR techniques is presented, which significantly improves the efficiency and accuracy of feature extraction by automatically extracting features from images and LiDAR data. Multi-source data fusion comprehensively captures the diversity and complexity of buildings, improving recognition accuracy and generalization ability. The innovation of this article lies in the combination of DCNN and LiDAR technology to solve the performance and accuracy problems of traditional methods, providing important data support and a decision-making basis for urban planning and management.

2. RELATED WORK

Urban building recognition is not a new research direction. As an inevitable topic in urban construction, there have been numerous related studies before. Early research mainly relied on traditional computer vision and image processing techniques, such as edge detection, texture analysis, and image segmentation. The edge detection algorithm proposed by Versaci M and the texture analysis method proposed by Chen D et al. achieved remarkable results in specific cases [7-8]. However, when confronted with a complex and changeable urban environment, especially when buildings are occluded, and affected by illumination changes or other environmental factors, these methods often exhibit low robustness and accuracy, leading to poor recognition performance [9]. With the advent of machine learning technology, researchers have attempted to use machine learning algorithms to improve the recognition of urban architectural landscapes. So far, traditional machine learning algorithms such as Support Vector Machine (SVM), decision trees, and K-Nearest Neighbors (k-NN), etc., have been widely applied in the field related to building recognition [10-11]. These algorithms automatically identify and classify buildings through model training. Compared with traditional image processing techniques, they show obvious advantages in processing large-scale data and complex pattern recognition tasks. Since traditional machine learning training relies on manually extracted feature vectors, their quality significantly affects the model performance. Moreover, in view of the complex and diverse characteristics of urban architectural landscapes, it is difficult to be comprehensive and precise through manual feature extraction, resulting in a significant decrease in the model recognition effect [12-13]. With the continuous progress of data collection technology, LiDAR technology has entered the field of urban architectural landscape recognition. Through emitting laser beams and receiving reflected signals, LiDAR technology can obtain high-precision 3D point cloud data containing information such as building height, volume, and shape [14]. Compared with traditional 2D image data, LiDAR data provides more abundant and detailed spatial information, significantly improving the accuracy and reliability of building recognition. Some researchers have conducted relevant studies. For example, Ji S et al. combined LiDAR data with aerial image data for automatic urban building extraction. However, these methods still face challenges in data fusion and feature extraction and have not fully exploited the advantages of multi-source data [15].

When researchers delve deeply into the potential applications of machine recognition, some are conducting application research on deep learning in the recognition field. Affected by DCNN, the field of image recognition has reached a new stage, and building recognition has benefited greatly from it. Through multiple convolution processes, DCNN can extract multi-level features by itself, enhancing the accuracy and efficiency of feature extraction [16]. In dealing with large amounts of data and complex patterns, the performance of DCNN is superior to traditional manual techniques, ensuring robustness and achieving the automation of feature extraction [17]. These characteristics have promoted academic research on the recognition of urban architectural landscapes based on DCNN. For example, Xu G accurately classified buildings in urban aerial images by virtue of the deep convolution network [18]. Nagy B enhanced the accuracy of model recognition by combining DCNN with LiDAR data [19]. The specific operation is to apply CNN to image data and 3D-CNN to LiDAR point cloud data and then conduct feature fusion, which can improve the accuracy of model recognition and classification [20]. Some researchers are delving into advanced technologies that can be applied to the recognition and reconstruction of urban building models, such as Generative Adversarial Networks (GANs) and autoencoders. Wang Z used GANs to construct high-resolution images for urban planning and design [21]. To achieve efficient data transmission and storage, Wang S et al. used autoencoders to compress and reconstruct 3D models of urban buildings [22]. LiDAR and DCNN technologies have great potential in identifying urban architectural landscapes. However, there are still problems with this technology that require more research and development.

3. PROCESSES

3.1 Data Collection and Data Preprocessing

To enhance the generalization ability of the model to the complexity and diversity of real image data, this article collects a large number of satellite images and point cloud data through datasets such as Google Maps as the experimental dataset, as shown in Figure 1. The ArcGIS software is used to interpret satellite images and classify and annotate buildings visually. The annotation process includes the drawing of building contours and the allocation of category labels, ensuring that each building is accurately labeled as a separate entity. For the convenience of training deep learning models in the future, the annotated dataset is divided into training, test, and validation sets in a ratio of 7:2:1. The training set is used for training and parameter optimization of the model; the test set is used for the final model performance evaluation; the validation set is responsible for model selection [23]. The preliminary formation of the urban architectural layout dataset is illustrated in Figure 2.

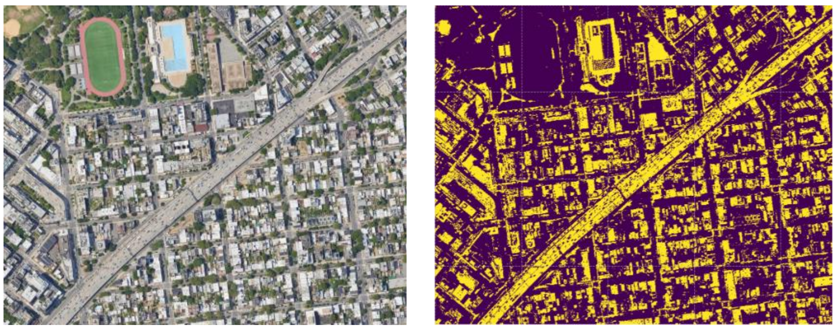

Figure 1: Collected urban images.

Figure 2: Urban view.

After the dataset partitioning is completed, in order to optimize data quality and improve the effectiveness of subsequent model training and evaluation, it is necessary to preprocess the data. Firstly, LiDAR data is denoised, and the statistical outlier removal method is used. Based on the neighborhood distance and height differences around each point, anomalous points in the LiDAR point cloud are identified and removed, improving the spatial consistency and accuracy of the data. By setting a threshold for the number of points within a fixed radius, low-density and abnormally distributed LiDAR point cloud data are filtered out, further cleaning the dataset [24]. Afterward, point cloud segmentation is performed on the denoised LiDAR data to extract the required point cloud information. This article adopts a segmentation algorithm based on region growth, which gradually merges adjacent points to form continuous building fragments based on the geometric and geographical attributes of point clouds, accurately reflecting the shape and contour of buildings [25], as shown in Figure 3. In order to extract features from LiDAR data, this article uses the RANdom SAmple Consensus (RANSAC) algorithm [26]. The initial iteration count is set to 100, and the distance threshold is 0.1 meters. At each iteration, 5 points in the dataset are randomly selected as sample points to calculate the relevant parameters of the plane model. Then, the distances from all data points to the fitting plane are calculated and points with distances less than the distance threshold are marked as interior points. The iteration stops when the previously set number of iterations is reached. The images before and after processing are shown in Figure 4.

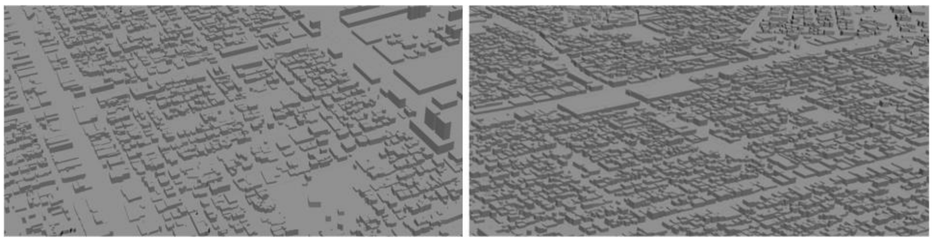

Figure 3: Building outlines under the region-growing segmentation algorithm.

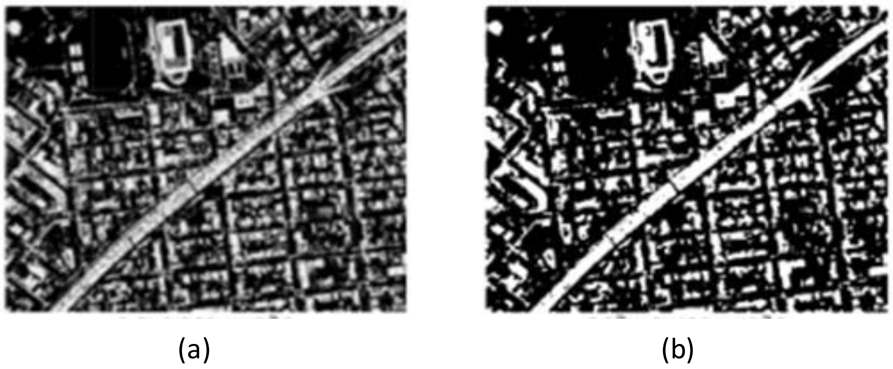

Figure 4: Image data after denoising and segmentation: (a) preprocessing image, (b) processed image.

In addition to processing LiDAR data, satellite images also need to be processed. Firstly, using GIS tools, the required areas are cropped to appropriate sizes, and bilinear interpolation is used to resample the cropped image, ensuring that all input images have the same spatial resolution. Due to the frequent influence of the atmosphere on satellite images, it is necessary to use Quick Atmospheric Correction (QUAC) tools to correct the images [27]. Through histogram equalization and Gaussian filtering, the contrast and noise removal of the image are enhanced. Afterwards, principal component analysis (PCA) is performed on the image to fuse some features and reduce redundant information [28]. Finally, similar to LiDAR data processing, feature extraction is required for the image. Urban planning feature extraction is shown in Figure 5. This article uses pre-trained ResNet-34 to extract advanced features. This model has fewer layers, fast inference speed, and can extract features that are detailed enough. In this study, the types of urban building recognition are divided into low-rise residential buildings, apartment-style residential buildings, and other types, and the above feature extraction is also based on these classifications.

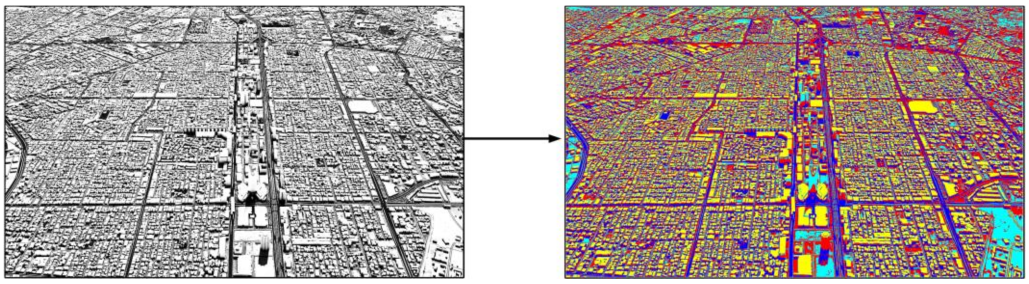

Figure 5: Urban planning feature extraction.

After the two types of data processing are completed, data registration is required to ensure that these two data sources can be aligned in the same geographic coordinate system, ensuring accurate spatial correspondence of multi-source data, and effectively combining and utilizing their respective advantageous features. The first step is to perform coarse feature registration. In this regard, this article adopts the commonly used nearest neighbor matching algorithm in feature matching. For the feature vectors l_i in LiDAR data and s_j in satellite image data, Euclidean distance is used to measure their similarity, as shown in Formula (1). The feature l_i in each LiDAR data can find the closest feature vector in satellite image data. In order to ensure the accuracy of matching, the distance Dmax and consistency threshold are set. When the distance between two vectors is less than Dmax, the matching is considered successful, and each vector l_i can best match up to 2 s_j [29].

$$d(l_i, s_j) = \sqrt{\sum_{k=1}^{n}(l_{i,k} - s_{j,k})^2} \tag{1}$$

After the rough matching is completed, it is necessary to perform a more precise matching. The Iterative Closest Point (ICP) algorithm used in this article is to find the optimal transformation between two sets of data through iterative optimization, aligning their overlapping parts as much as possible [30]. At the beginning, the preliminary transformation relationship between the two sets of data is assumed. By searching for nearest neighbors, a point correspondence between two sets of data is established. Based on the point correspondence obtained above, the optimal transformation for minimizing the error of two sets of data is calculated. The calculated optimal transformation is applied to one set of data and its position is updated. Calculations, updates, and transformations are continuously performed until the changes are minimal. Afterward, by using overlay display, transparency adjustment, and other methods, the matching degree between the registration results and the original data is compared. Finally, the stability and consistency of the registration results are verified through multiple registration experiments.

The registered data still needs to be fused into a joint feature map before it can be used for model training. This step of this article adopts a self-attention mechanism. Firstly, the features of the data are linearly transformed to obtain the corresponding query matrix Q, key matrix K, and value matrix V. The specific transformation formulas are shown in Formulas (2), (3), and (4):

$$Q = SW_Q \tag{2}$$

$$K = LW_K \tag{3}$$

$$V = LW_V \tag{4}$$

Among them, W_Q, WK, W_V are the weight matrices for converting satellite image features into query vectors, and LiDAR features into key vectors and value vectors, respectively. Afterward, it is necessary to measure the correlation between image features and LiDAR features, which determines which LiDAR features are important to a certain image feature. The specific operation is to calculate the similarity between the query vector and the key vector through the dot product and use the softmax function to obtain the corresponding attention weight matrix. The calculation formula is shown in Formula (5). Matrix A contains the similarity between each query and all keys, representing the similarity between specific image features and all LiDAR features [31].

$$A = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) \tag{5}$$

After obtaining matrix A, the vector V needs to be weighted and summed with A, as shown in Formula (6). Among them, Mi is the i-th fused feature. The purpose of this weighted sum is to emphasize the most important part of LiDAR features in satellite image features through the weight matrix A.

$$M_i = \sum_{j=1}^{N} A_{ij} V_j \tag{6}$$

Urban planning before and after data registration is shown in Figure 6.

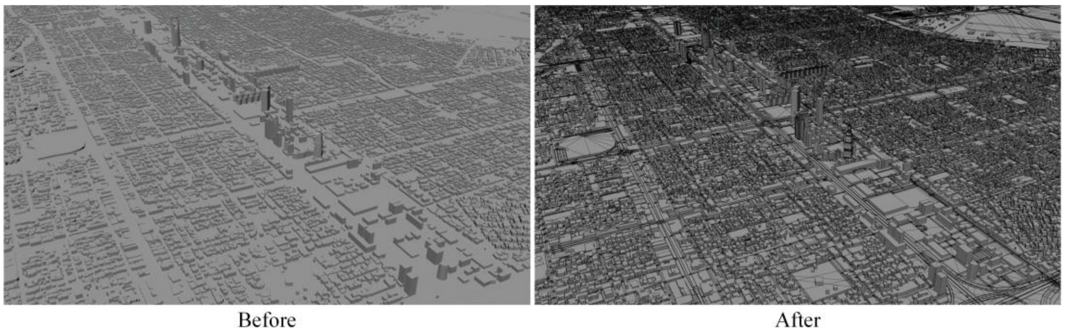

Figure 6: Urban planning before and after data registration.

3.2 Model Training

Before starting model training, the first step is to select deep learning models. At the beginning of this article, the following four models are considered: VGG (Visual Geometry Group) 16, which has 16 layers of depth and has a fast processing speed for medium-sized images, but with a large number of parameters and high computational resources; ResNet50, with a depth of 50 layers and an application of a residual block structure to solve the gradient vanishing problem in the network; InceptionV3, having parallel convolutional and pooling layers of different sizes, and maintaining high precision while reducing computational complexity through separating convolutions; DenseNet121, with a depth of up to 121 layers, which greatly enhances feature transfer and reuse through dense connections, and with a small number of parameters and high computational efficiency [32]. From the basic characteristics of the model, it can be seen that each model has its own characteristics. In order to select the most suitable model, this article uses a validation set to conduct simple training and evaluation on the four models mentioned above, and the results obtained are shown in Table 1. Through a comprehensive comparison of accuracy, Mean Square Error (MSE), and F1 score, ResNet50 performs the best and can be used as the final model in this article.

Model	Accuracy	MSE	F1 Score
VGG16	0.88	0.15	0.87
ResNet50	0.93	0.10	0.92
InceptionV3	0.90	0.13	0.89
DenseNet121	0.91	0.12	0.90

Table 1: Accuracy, MSE, and F1 score of each model.

After selecting the model, relevant initialization settings are carried out. In this article, He initialization is chosen, which performs well in convolutional neural networks using ReLU activation functions. The first step in initialization is to determine the number of input nodes, calculated based on the convolutional layer size of kxk and the input channel of C. The number of input nodes is:

$$n_{\text{in}} = k \times k \times C \tag{7}$$

For fully connected layers, the number of input nodes is the number of output nodes from the previous layer, according to the standard deviation formula:

$$\sigma = \sqrt{\frac{2}{n_{\text{in}}}} \tag{8}$$

After calculating the corresponding standard deviation, random weights are generated through a normal distribution $w \sim N(0, \sigma^2)$. This can generate reasonable initial values for the weight matrix of each layer, ensuring that the model can smoothly perform gradient descent and quickly converge to the optimal solution in the early stages of training.

In terms of training parameters, Table 2 can be obtained by conducting relevant experiments on the validation set while ensuring other conditions are the same. When the number of training rounds is about 50, the loss values on the validation set tend to stabilize, indicating that the model has reached a good convergence state. When the batch size is 32, the model ensures high accuracy while also keeping the training time within a reasonable range. When the learning rate is 0.001, the model converges quickly and the loss value steadily decreases.

Parameter	Value	Validation Loss	Training Time (s/epoch)	Accuracy (%)
Epochs	10	0.47	-	-
	20	0.35	-	-
	30	0.28	-	-
	40	0.24	-	-
	50	0.22	-	-
Batch Size	16	-	50	91.2
	32	-	45	92.5
	64	-	40	91.8
Learning Rate	0.01	0.25	-	-
	0.001	0.19	-	-
	0.0001	0.22	-	-

Table 2: Effects of training rounds, different batches, and learning rates on training.

In terms of defining the loss function of the model, classification cross-entropy is used, such as Formula (9):

$$L_i = -\frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T} y_{n,t}\, log(\hat{y}_{n,t}) \tag{9}$$

Among them, N is the number of samples; T is the number of categories; $y_{n,t}$ represents the true label of sample n in category c; $\hat{y}_{n,t}$ represents the predicted probability of sample i in category c. In order to overfit the model, this article uses L2 regularization to restrict larger weight values and make the model parameters smoother. The formula is shown in Formula (10). Among them, L is the original loss function; λ is the regularization strength hyperparameter; w_i is the weight parameter [33].

$$L_{regi} = L_i + \lambda \sum_{i=1}^{n} w_i^2 \tag{10}$$

When selecting the optimizer, this article also conducts a data comparison. Under three different optimizer conditions: SGD (stochastic gradient descent), RMSProp (Root Mean Square Propagation), and Adam, experiments are conducted on the validation set, and the corresponding results are shown in Figure 7. The Adam optimizer has the lowest validation set loss value, the same training time as RMSProp, and a final accuracy of 92.5%, which is the highest among the three optimizers. It can be chosen as the final optimizer studied in this article.

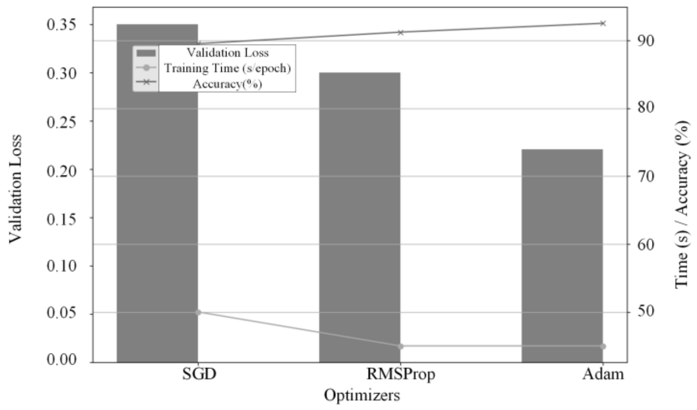

Figure 7: The impact of different optimizers on training.

In addition to regularizing the function, it is also necessary to add a Dropout layer after the fully connected layer of the model to further reduce the dependency relationships between neurons and prevent overfitting.

Once the initial parameters of the model are set, formal model training needs to be carried out. Firstly, the data in the training set is divided into small batches, each with 32 samples. Random cropping, flipping, and other operations are performed on the data to increase the diversity of the samples, which helps to improve the model's generalization ability. Afterward, forward propagation is carried out, which is the process of obtaining the predicted result $\hat{y}$ from input data I through model F. In this process, the input data undergoes a series of linear transformations and nonlinear activation functions and is transmitted layer by layer to the output layer to obtain the model's predictions for each sample, as shown in Formula (11). Among them, W is the model weight.

$$\hat{y} = F(I; W) \tag{11}$$

There are many levels in model F, and after expanding the previously obtained fused feature map into a one-dimensional vector, it can be input into a fully connected layer. This layer fuses various local features globally to form a global description of the entire image and performs nonlinear changes by mapping them into the ReLU activation function to fit more complex functions.

Assuming the output of the fully connected layer is A, A needs to classify the model through the SoftMax layer. This layer converts the output of the fully connected layer into the probability distribution of each category, exponentiates, and normalizes the original score so that the sum of probabilities of all categories is 1. This allows for a clear understanding of the model's prediction results, as shown in Formula (12). Among them, T is the total number of classifications.

$$p_i = \frac{e^{A_i}}{\sum_{j=1}^{T} e^{A_j}} \tag{12}$$

Afterward, the loss between the predicted results and the true labels is calculated using the previously defined loss function, and the gradient of the loss function on the weight W is calculated through backpropagation, as shown in Formula (13). Among them, $\frac{\partial L}{\partial \hat{y}}$ is the gradient of the loss function on the predicted value.

$$\frac{\partial L}{\partial W} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial W} \tag{13}$$

In addition to regularizing the loss function, it is also necessary to add a dropout layer after the fully connected layer of the model to ensure that the model does not overfit [34]. After joining this layer, during the forward propagation process, the Dropout layer randomly sets the output of some neurons to 0, preventing these neurons from participating in the calculation in the current iteration. These discarded neurons do not have an impact on subsequent layers. For neurons that have not been discarded, their output is scaled up proportionally to maintain the expected value of the overall output. The formula for the internal connections of forward propagation is shown in Formula (14):

$$o_i^{(l)} = \begin{cases} 0 & p \\ \frac{o_i^{(l)}}{1-p} & 1-p \end{cases} \tag{14}$$

Among them, $o_i^{(l)}$ is the output of the i-th neuron in the l-th layer, and p is the Dropout probability. During the backpropagation process, only neurons that have not been discarded participate in gradient calculation. The Dropout layer sets the gradient of discarded neurons to zero to prevent them from affecting the update of model parameters. In the subsequent verification testing phase, the Dropout layer is closed, meaning that all neurons participate in the calculation and the output is not randomly discarded. In order to maintain consistency in output during training and testing, the Dropout layer does not perform any operations at this time, and the data passes directly through the layer.

So far, a training session has been completed, and the weights of the model need to be updated using an optimizer to calculate the gradients. The updated formula for gradients is shown in Formula (15), where ɑ is the learning rate. p_t and x_t are the momentum and variables of the optimizer, while ϵ is a constant used to ensure numerical stability.

$$W_{t+1} = W_t - \alpha \cdot p_t / (\sqrt{x_t} + \epsilon) \tag{15}$$

Throughout the entire training process, the above steps are constantly repeated, and after each training, the test set is used for evaluation to calculate the corresponding evaluation indicators. Model recognition of partial urban planning areas is shown in Figure 8.

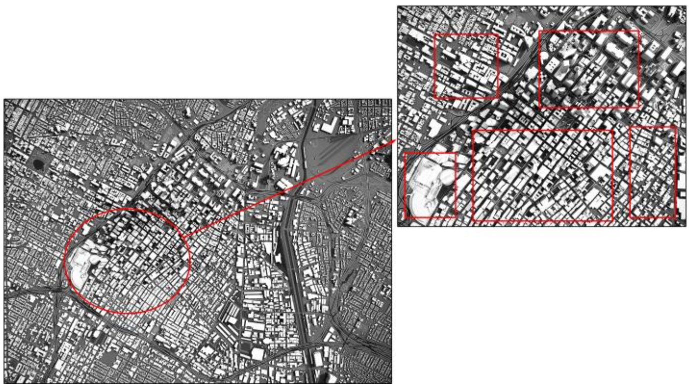

Figure 8: Urban planning area recognition.

4. RESULTS

In order to compare with traditional recognition models, this article uses the ResNet50 model without LiDAR data as the control group, which is evaluated experimentally using the same dataset as the model in this article. The learning curves of the two models throughout the entire training process are shown in Figure 9.

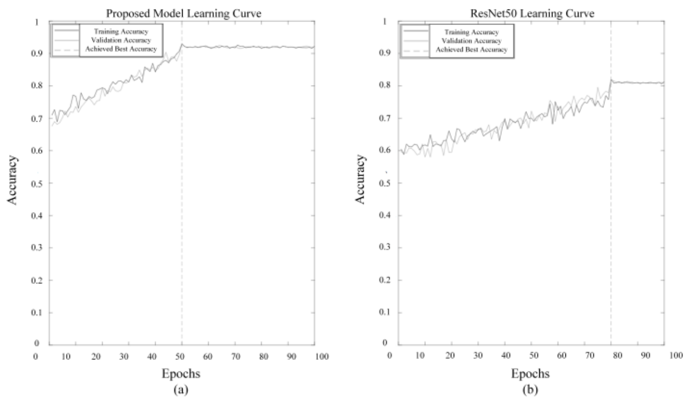

Figure 9: Learning curves of the model in this article and ResNet50 model: (a) model in this article, (b): ResNet50 model.

When analyzing the learning curves of the model in this article and the ResNet50 model, their dynamic changes and performance differences can be observed during the training process. The model in this article exhibits significant fluctuations in training and validation accuracy in the initial stage. This fluctuation is mainly due to the sensitivity of the model to data in the early learning stage and the influence of random initialization parameters. As the number of training iterations increases, the model gradually optimizes and the accuracy fluctuates, ultimately reaching an accuracy of about 93%. This indicates that the model in this article can achieve high predictive performance after a certain training period of learning and optimization. After achieving a certain level of accuracy, the learning curve of the model in this article shows a characteristic of high stability. Although there are still slight fluctuations after reaching a relatively stable accuracy, the overall trend remains at a high level, indicating that the model can stably maintain good predictive ability near the optimal parameters. This stability is crucial for the reliability and consistency of models in practical applications. In contrast, the ResNet50 model exhibits similar fluctuation characteristics, although it also experiences significant accuracy fluctuations in the initial stage. However, the final accuracy of the ResNet50 model is about 82%, which is lower than the model proposed in this article. The ResNet50 model shows good accuracy and stability in the later stage of training. Although the overall level is slightly lower, it still maintains acceptable predictive performance, but compared to the model in this article, it is slightly insufficient.

Accurately evaluating the performance of recognition classification models is crucial in urban landscape recognition research. During the testing process of the final model, a confusion matrix is generated by comparing the real labels with the predicted labels. This matrix not only demonstrates the accuracy of the model on each category but also reveals the confusion of the model between different categories. The confusion matrix obtained from the testing and evaluation of the two models is shown in Figure 10.

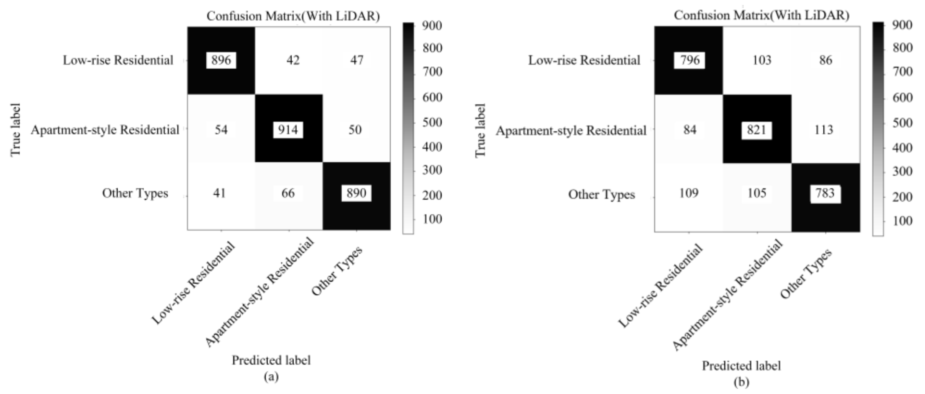

Figure 10: Confusion matrix between the model in this article and ResNet50 model: (a) confusion matrix of the model in this article, (b) confusion matrix of ResNet50 model.

The model in this article has a high recognition accuracy in low-rise residential samples. Specifically, 896 low-rise residential samples are correctly identified, much higher than the 796 for the ResNet50 model. This indicates that combining LiDAR data significantly improves the model's ability to recognize low-rise residential dwellings. Similarly, for apartment-style residential dwellings and other types, the number of correct identifications of this article's model is higher than that of the pure ResNet50 model. For apartment-style residential dwellings, the model in this article correctly identifies 914 samples, while the ResNet50 model identifies 821 samples. For other types, the model in this article identifies 890, while the pure ResNet50 model identifies 783. It can be concluded that the application of LiDAR data has played an important role in enriching image features and enhancing the model's capture of details. In contrast, the pure ResNet50 model shows a higher misclassification rate in all categories, further demonstrating the importance of LiDAR data.

Taking the stable model trained on the test set as the final test result, the corresponding accuracy, precision, recall, and F1 score can be obtained, as shown in Table 3.

Metric		Proposed Model	ResNet50 Model
	Accuracy	0.9310	0.8276
Pre cision	Low-rise Residential	0.9290	0.8329
	Apartment-style Residential	0.9174	0.8148
	Other Types	0.9241	0.8148
Rec all	Low-rise Residential	0.9177	0.7977
	Apartment-style Residential	0.9147	0.8080
	Other Types	0.9179	0.8076

F1 Score	Low-rise Residential	0.9233	0.8149
	Apartment-style Residential	0.9160	0.8114
	Other Types	0.9210	0.8112

Table 3: Accuracy, precision, recall, and F1 score corresponding to the two models

According to the data in Table 3, the model in this article demonstrates significant advantages over the ResNet50 model in all evaluation indicators. In terms of accuracy, the accuracy of the model in this article reaches 93.10%, while the ResNet50 model is only 82.76%. This significant difference directly reflects the effective improvement of model performance when combined with LiDAR data. In terms of precision, the model in this article achieves 92.90%, 91.74%, and 92.41%, respectively, in the classification of low-rise residential buildings, apartment-style residential buildings, and other types, while the ResNet50 model achieves 83.29%, 81.48%, and 81.48%, respectively. From the above data, it can be seen that the classification error of the model in this article is significantly smaller than that of the ResNet50 model, and its classification results are more reliable. From the perspective of recall rate, the recall rates of the three types of this model in this article are 91.77%, 91.47%, and 91.79%, respectively. Under the same conditions, the recall rates of the ResNet50 model are only 79.77%, 80.80%, and 80.76%. This reflects the superiority of the model in this article in correctly identifying positive samples and the effectiveness of the combination of multi-source data. From the perspective of the F1 score, for low-rise residential, apartment-style residential, and other types, the model in this article achieves 92.33%, 91.60%, and 92.1%, respectively, while the ResNet50 model achieves 81.49%, 81.14%, and 81.12%. This difference also demonstrates the advantages of the model in this article in terms of overall classification performance.

From the test set, the ROC curve (Receiver Operating Characteristic curve) and Precision-Recall curve (PR curve) shown in Figures 11 and 12 can also be obtained. In the ROC image of the model in this article, the AUC (Area Under Curve) of various buildings is around 0.96, with the highest AUC for low-rise residential buildings reaching 0.97. This demonstrates the high accuracy and robustness of the model in distinguishing these buildings in this article. In contrast, the image display performance of ResNet50 is relatively low, with an average AUC of around 0.84 and a maximum value of only 0.85. The AUC for all categories is lower than that of the model in this article. From the shape of the image curves, both models have curves that lean toward the upper left corner, indicating a lower false alarm rate and a higher hit rate. However, compared to this, the image on the right is closer to the diagonal, indicating poorer discrimination ability. From the PR curve graph, it can be seen that the PR curves of each classification in the model of this article are closer to the upper right corner than those of the ResNet50 model, indicating that the model in this article has higher precision and recall. In addition, it can be seen that the area below the curves of each classification in this article's model is larger than that of the ResNet50 model, which is basically above the curve of the ResNet50 model. This indicates that the performance of this article's model is better than that of the ResNet50 model.

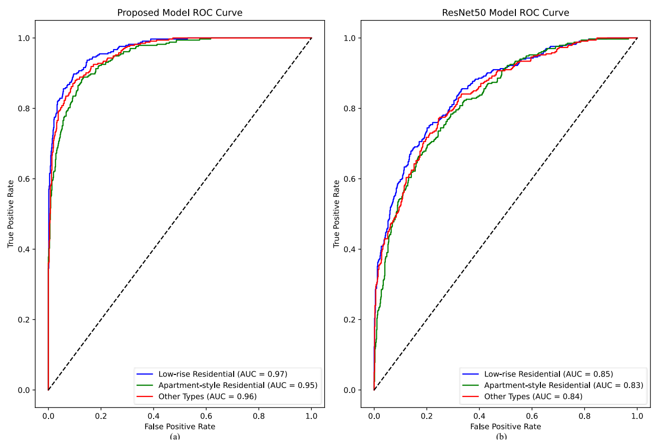

Figure 11: ROC images of the model in this article and ResNet50 model: (a) ROC image of the model in this article, (b) ROC image of ResNet50 model.

5. CONCLUSIONS

This article studied the efficient identification and classification of urban architectural landscapes through the combination of DCNN and LiDAR data. By comparing the experimental results, it can be seen that using LiDAR data can significantly improve the recognition accuracy of the model in low-rise residential buildings, apartment-style residential buildings, and other building types, with lower training time costs. LiDAR data not only provides precise spatial information but also helps the model better distinguish between buildings and the surrounding environment. Overall, the combination of deep convolutional networks and LiDAR data not only improves the precision and speed of urban architectural landscape recognition but also provides valuable method references and experimental benchmarks for similar research in the future.

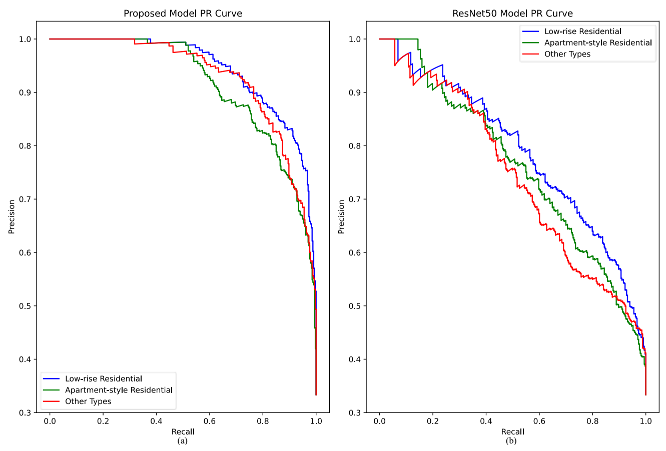

Figure 12: PR curve of the model in this article and ResNet50 model: (a) PR curve of the model in this article, (b)): PR curve of ResNet50 model.

Xiaoyong Zhang, https://orcid.org/0009-0006-3460-498X
Shan Mi, https://orcid.org/0009-0000-6435-7533
Xiuhuan Feng, https://orcid.org/0009-0009-3512-0620
Weiying Ding, https://orcid.org/0009-0001-8523-5751

REFERENCES

[1] Zebari, R.; Abdulazeez, A.; Zeebaree, D.; Zebari, D.; Saeed, J.: A comprehensive review of dimensionality reduction techniques for feature selection and feature extraction, Journal of Applied Science and Technology Trends, 1(1), 2020, 56-70. https://doi.org/10.38094/jastt1224
[2] Vargas-Munoz, J. E.; Srivastava, S.; Tuia, D.; Falcão, A.: OpenStreetMap: Challenges and opportunities in machine learning and remote sensing, IEEE Geoscience and Remote Sensing Magazine, 9(1), 2020, 184-199. https://doi.org/10.1109/MGRS.2020.2994107
[3] Deng, T.; Zhang, K.; Shen, Z. J. M.: A systematic review of a digital twin city: A new pattern of urban governance toward smart cities, Journal of Management Science and Engineering, 6(2), 2021, 125-134. https://doi.org/10.1016/j.jmse.2021.03.003
[4] Gao, B.; Pan, Y.; Li, C.; Geng, S.: Are we hungry for 3D LiDAR data for semantic segmentation? A survey of datasets and methods, IEEE Transactions on Intelligent Transportation Systems, 23(7), 2021, 6063-6081. https://doi.org/10.1109/TITS.2021.3076844
[5] Ilesanmi, A. E.; Ilesanmi, T. O.: Methods for image denoising using convolutional neural network: a review. Complex & Intelligent Systems, 7(5), 2021, 2179-2198. https://doi.org/10.1007/s40747-021-00428-4
[6] Shen, M.; Tang, X.; Zhu, L.; Du, X.; Guizani, M.: Privacy-preserving support vector machine training over blockchain-based encrypted IoT data in smart cities, IEEE Internet of Things Journal, 6(5), 2019, 7702-7712. https://doi.org/10.1109/JIOT.2019.2901840
[7] Versaci, M.; Morabito, F. C.: Image edge detection: A new approach based on fuzzy entropy and fuzzy divergence, International Journal of Fuzzy Systems, 23(4), 2021, 918-936. https://doi.org/10.1007/s40815-020-01030-5
[8] Chen, D.: Evaluating asphalt pavement surface texture using 3D digital imaging, International Journal of Pavement Engineering, 21(4), 2020, 416-427. https://doi.org/10.1080/10298436.2018.1483503
[9] Hou, Y.; Li, Q.; Zhang, C.; Lu, G.; Ye, Z.; Chen, Y.; et al.: The state-of-the-art review on applications of intrusive sensing, image processing techniques, and machine learning methods in pavement monitoring and analysis, Engineering, 7(6), 2021, 845-856. https://doi.org/10.1016/j.eng.2020.07.030
[10] Yu, C.; Chen, J.: Landslide susceptibility mapping using the slope unit for southeastern Helong City, Jilin Province, China: a comparison of ANN and SVM, Symmetry, 12(6), 2020, 1047. https://doi.org/10.3390/sym12061047

[11] Guan, Y.; Lu, R.; Zheng, Y.; Shao, J.; Wei, G.: Toward oblivious location-based k-nearest neighbor query in smart cities, IEEE Internet of Things Journal, 8(18), 2021, 14219-14231. https://doi.org/10.1109/JIOT.2021.3068859

[12] Chen, Q.; Wang, W.; Wu, F.; De, S.; Wang, R.; Zhang, B.; et al.: A survey on an emerging area: Deep learning for smart city data, IEEE Transactions on Emerging Topics in Computational Intelligence, 3(5), 2019, 392-410. https://doi.org/10.1109/TETCI.2019.2907718

[13] Li, X.; Li, C.; Rahaman, M. M.; Sun, H.; Li, X.; Wu, J.; et al.: A comprehensive review of computer-aided whole-slide image analysis: from datasets to feature extraction, segmentation, classification and detection approaches, Artificial Intelligence Review, 55(6), 2022, 4809-4878. https://doi.org/10.1007/s10462-021-10121-0

[14] dos Santos, R. C.; Galo, M.; Carrilho, A. C.: Extraction of building roof boundaries from LiDAR data using an adaptive alpha-shape algorithm, IEEE Geoscience and Remote Sensing Letters, 16(8), 2019, 1289-1293. https://doi.org/10.1109/LGRS.2019.2894098

[15] Ji, S.; Wei, S.; Lu, M.: A scale robust convolutional neural network for automatic building extraction from aerial and satellite imagery, International Journal of Remote Sensing, 40(9), 2019, 3308-3322. https://doi.org/10.1080/01431161.2018.1528024

[16] Cai, Z.; Vasconcelos, N.: Cascade R-CNN: High-quality object detection and instance segmentation, IEEE Transactions on Pattern Analysis and Machine Intelligence, 43(5), 2019, 1483-1498. https://doi.org/10.1109/TPAMI.2019.2956516

[17] Ahmed, S. F.; Alam, M. S. B.; Hassan, M.; Rozbu, M. R.; Ishtiak, T.; Rafa, N., et al.: Deep learning modelling techniques: current progress, applications, advantages, and challenges, Artificial Intelligence Review, 56(11), 2023, 13521-13617. https://doi.org/10.1007/s10462-023-10466-8

[18] Xu, G.; Zhu, X.; Tapper, N.; Bechtel, B.: Urban climate zone classification using convolutional neural network and ground-level images, Progress in Physical Geography: Earth and Environment, 43(3), 2019, 410-424. https://doi.org/10.1177/0309133319837711

[19] Nagy, B.; Benedek, C.: 3D CNN-based semantic labeling approach for mobile laser scanning data, IEEE Sensors Journal, 19(21), 2019, 10034-10045. https://doi.org/10.1109/JSEN.2019.2927269

[20] Kumar, B.; Pandey, G.; Lohani, B.; Misra, S.: A framework for automatic classification of mobile LiDAR data using multiple regions and 3D CNN architecture, International Journal of Remote Sensing, 41(14), 2020, 5588-5608. https://doi.org/10.1080/01431161.2020.1734252

[21] Wang, Z.; Xu, N.; Wang, B.; Liu, Y.; Zhang, S.: Urban building extraction from high-resolution remote sensing imagery based on multi-scale recurrent conditional generative adversarial network, GIScience & Remote Sensing, 59(1), 2022, 861-884. https://doi.org/10.1080/15481603.2022.2076382

[22] Wang, S.; Guo, J.; Zhang, Y.; Hu, Y.; Ding, C.; Wu, Y.: Tomosar 3d reconstruction for buildings using very few tracks of observation: A conditional generative adversarial network approach, Remote Sensing, 13(24), 2021, 5055. https://doi.org/10.3390/rs13245055

[23] Helber, P.; Bischke, B.; Dengel, A.; Borth, D.: Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification, IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, 12(7), 2019, 2217-2226. https://doi.org/10.1109/JSTARS.2019.2918242

[24] Liu, H.; Li, J.; Wu, Y.; Fu, Y.: Clustering with outlier removal, IEEE Transactions on Knowledge and Data Engineering, 33(6), 2019, 2369-2379. https://doi.org/10.1109/TKDE.2019.2954317

[25] Lao, J.; Wang, C.; Nie, S.; Xi, X.; Long, H.; Feng, B.: A new denoising method for photon-counting LiDAR data with different surface types and observation conditions, International Journal of Digital Earth, 16(1), 2023, 1551-1567. https://doi.org/10.1080/17538947.2023.2203952

[26] Gonultaş, F.; Atik, M. E.; Duran, Z.: Extraction of roof planes from different point clouds using RANSAC algorithm, International Journal of Environment and Geoinformatics, 7(2), 2020, 165-171. https://doi.org/10.30897/ijegeo.715510

[27] Liu, N.; Zou, B.; Feng, H.; et al.: Evaluation and comparison of the multiangle implementation of the atmospheric correction algorithm, Dark Target, and Deep Blue aerosol products over China, Atmospheric Chemistry and Physics, 19(12), 2019, 8243-8268. https://doi.org/10.5194/acp-19-8243-2019

[28] Uddin, M. P.; Mamun, M. A.; Hossain, M. A.: PCA-based feature reduction for hyperspectral remote sensing image classification, IETE Technical Review, 38(4), 2021, 377-396. https://doi.org/10.1080/02564602.2020.1740615

[29] Isnain, A. R.; Supriyanto, J.; Kharisma, M. P.: Implementation of K-Nearest Neighbor (K-NN) algorithm for public sentiment analysis of online learning, IJCCS (Indonesian Journal of Computing and Cybernetics Systems), 15(2), 2021, 121-130. https://doi.org/10.22146/ijccs.65176

[30] Lee, J.; Lee, I.; Kang, J.: A Study on Position Matching Technique for 3D Building Model using Existing Spatial Data-Focusing on ICP Algorithm Implementation, Journal of Cadastre & Land InformatiX, 51(1), 2021, 67-77. https://doi.org/10.22640/lxsiri.2021.51.1.67

[31] Ye, M.; Shen, J.; Zhang, X.; Yuen, P.; Chang, S.: Augmentation invariant and instance spreading feature for softmax embedding, IEEE Transactions on Pattern Analysis and Machine Intelligence, 44(2), 2020, 924-939. https://doi.org/10.1109/TPAMI.2020.3013379

[32] Khan, A.; Chefranov, A.; Demirel, H.: Building discriminative features of scene recognition using multi-stages of inception-ResNet-v2, Applied Intelligence, 53(15), 2023, 18431-18449. https://doi.org/10.1007/s10489-023-04460-4

[33] Zadeh, S. G.; Schmid, M.: Bias in cross-entropy-based training of deep survival networks, IEEE Transactions on Pattern Analysis and Machine Intelligence, 43(9), 2020, 3126-3137. https://doi.org/10.1109/TPAMI.2020.2979450

[34] Garbin, C.; Zhu, X.; Marques, O.: Dropout vs. batch normalization: an empirical study of their impact to deep learning, Multimedia Tools and Applications, 79(19), 2020, 12777-12815. https://doi.org/10.1007/s11042-019-08453-9