scientific reports

Check for updates

OPEN

Adaptive deep learning-based neighborhood search method for point cloud

Qian Xiang[1], Yuntao He[1]✉ & Donghai Wen[2]

Point cloud processing is a highly challenging task in 3D vision because it is unstructured and unordered. Recently, deep learning has been proven to be quite successful in point cloud recognition, registration, segmentation, etc. Neighborhood search operation is an important component of point cloud deep learning models, and directly affects the performance of the model. In this paper, we propose a learnable neighborhood search method. This method adaptively chooses an appropriate search method based on the characteristics of each point, thus avoiding the disadvantage of selecting the search method manually. We validate the proposed methods on ModelNet40 dataset and ShapeNetPart dataset, and all the chosen models achieved a performance improvement with a maximum improvement of 1.1%. The proposed method is a plug-and-play technique and can be easily integrated into existing methods.

The rapid development of 3D sensors has increased the demand for point cloud processing technology. At present, point cloud processing technology is widely used in sensor systems such as AR, autonomous driving, and pose estimation[1–3] In recent years, deep learning methods are widely used in point cloud processing and have achieved good results[4–6]. Because a single point cannot provide enough information to identify a local structure, neighborhood search technology is very important in point cloud analysis. At present, there are two main methods to achieve the objective: one is to search a neighborhood in 3D space, and the other is to search it in feature space.

The difference between the two methods lies in their search space. The former method searches in 3D space, while the latter method searches in feature space. This leads to the formation of different neighborhoods by the two methods. The former method aggregates local information, thus preserving the local topological relationship. In this way, a point cloud is divided into a group of 1-hop subgraphs. In the latter method, proximity in feature space differs from proximity in 3D space, leading to the nonlocal diffusion of information throughout the point cloud. The neighborhood in feature space is equivalent to a group of dynamic n-hop subgraphs from 3D space, which gives the feature space a stronger ability to capture global features. Especially in multilayer systems, affinity in feature space captures semantic characteristics over potentially long distances in the original embedding.

The neighborhood searched in 3D space has clear local topology information, which makes it widely used in point cloud normal estimation, feature extraction, outlier removal and other tasks; examples include PointNet++[7], PointCleanNet[8] and D3Feat[9]. Because the neighborhood searched in feature space makes better use of learned features, it has rich semantic information and makes it easier to obtain global information. Therefore, it has a good effect on unsupervised learning, feature retrieval and semantic segmentation, as in GraphTER[10] DGCNN[11], and LDGCNN[12]. Many studies have been devoted to improving the flexibility of 3D space searches, such as specifying the search direction[13] and clustering points to special points[14] There have also been some studies on building a better feature neighborhood, such as using clustering instead of brute search. Using a data structure to speed up the search is an important improvement direction, such as using KD-Tree[15]or OC-Tree[16] to speed up the search.

Generally, 3D space searching only depends on local information, so the information it provides is not rich enough, and selecting the neighborhood size is difficult. Feature space searching obtains enough local and global information due to the introduction of long dependency, but the features are not abstract enough in the low-level layers in multilayer networks, which often leads to the neighborhoods in feature space being unstable and having no clear meaning.

For the problem of how to select the search method in multilayer networks, one option is to use 3D space searching to obtain local neighborhoods and aggregate features in the low-level layers and use feature space

[1]School of Electronic and Information Engineering, Beihang University, Beijing 100191, China. [2]Troops 66133, PLA, Beijing 100191, China. ✉email: yuntaohe@buaa.edu.cn

searching to obtain nonlocal neighborhoods in the high-level layers to enrich the extracted information. This method has been proven to be feasible in experiments[17], but to the best of our knowledge, there is no clear design principle to find the best search method directly according to different network attributes.

Our approach is to change the selection of the search method into parameters that can be optimized by the model. The method links a neighborhood in 3D space with a neighborhood in feature space. In addition, we assign weights to these two branches that are learned from a point attention feature. We call this "soft-shortcut" ATSearch. Thus, each point can adjust its search tendency adaptively according to its own characteristics.

In summary, the main achievements of this paper are follows:

(1) In view of the characteristics of the existing 3D point cloud neighborhood search methods, this paper proposes an adaptive neighborhood search method based on point features and attention module.
(2) The proposed method is plug-and-play and can be easily inserted into the existing point cloud deep learning models in many tasks.
(3) The performances of proposed method are evaluated on popular 3D shape classification and segmentation data sets. Experimental results show that the proposed method can improve the performance of the models.

The rest of this paper is organized as follows. The "Methods" section introduces the method of the proposed neighborhood search method based on the features of point cloud and different search spaces. The "Results" section verified the effectiveness of the proposed method through experiments. The "Discussion" section studied the proposed search method, robustness-test, visualization and ablation experiments are performed in this section. "Conclusion" are in the last section.

Methods

Traditional neighborhood search methods for point cloud. In previous works, the neighborhood searching methods for point cloud can be categorized into two ones, one method is based on the spatial distance between pairs of points, which finds the neighbor points in 3D space, shown in (1). The other method is based on the feature distance between pairs of points, shown in (2).

$$\begin{cases} \mathrm{d_{xyz}} = \left\| p_i - p_j \right\|, \forall p_j \in P \\ N(p_i) = \{p^j | d^j_{xyz} < d_k\} \end{cases} \tag{1}$$

$$\begin{cases} \mathrm{d_f} = \left\| f_i - f_j \right\|, \forall f_j \in F \\ N(\mathrm{f_i}) = \{\mathrm{f}^j | d^j_f < d_k\}, i = 1, \ldots, k \end{cases} \tag{2}$$

where $\mathrm{P} \in \mathbb{R}^{N\times 3}$ and $F \in \mathbb{R}^{N\times C}$ represent the positions and features of points in the point cloud. $d_{xyz} \in \mathbb{R}^{N\times N}$, and $d_f \in \mathbb{R}^{N\times N}$ represent the pairwise distances of points in 3D space and in feature space, respectively. $p_i \in \mathbb{R}^3$ and $f_i \in \mathbb{R}^C$ represent the center points, and $p_j \in \mathbb{R}^3$ and $f_j \in \mathbb{R}^C$ are the corresponding neighbor points.

Search methods based on the spatial distance focus on preserving local information, and search methods based on feature distance focus on global information and nonlocal diffusion of information. In point cloud deep learning models, different convolution layers have different requirements for local and nonlocal information[17]. Therefore, manually selecting a search method will limit the performance of the models.

Proposed adaptive search method for point cloud. Equation (1) and (2) show that the key process of the traditional search method is to calculate the distance between points, In order to obtain the advantages of search methods in 3D or feature space, we propose a new method: attention search (ATSearch), which uses a "soft" shortcut to account for the feature distance and spatial distance when searching neighborhoods. The search method proposed in this paper is a combination of the advantages of the search methods in 3D space and feature space, which is essentially a secondary sampling from the 3D neighborhood based on 3D coordinate distance and the neighborhood in feature space based on the feature distance. So that both local and nonlocal information can be obtained to further improve the model performances. The differences from traditional search methods are shown in Fig. 1.

$$\begin{cases} d_{mix} = w_{xyz} \cdot d_{xyz} + w_f \cdot d_f \\ N(p_i) = \{p^i | d^i_{mix} < d_k\}, i = 1, \ldots, k \end{cases} \tag{3}$$

where $w_{xyz} \in \mathbb{R}^N$ and $w_f \in \mathbb{R}^N$ represent the point search preference. It can be easily seen that (1) and (2) are special cases of (3) for $w_{xyz} = 1, w_f = 0$ and $w_{xyz} = 0, w_f = 1$, respectively. Taking the general case, we can obtain a "mixed" distance d_{mix} by weighting d_{xyz} and d_f and then searching the neighborhood on it. The method proposed in this paper can search neighborhoods more flexibly, and it can not only choose how to search neighborhoods according to the characteristics of a point but also enhance the model's ability to find features across regions. The selection of w_{xyz} and w_f needs to be considered carefully for good performance.

To determine the selection range of w_{xyz} and w_f, we first define their basic rules to ensure that the selection is logical: (1) $w_{\mathrm{xyz}}, w_{\mathrm{f}} \geq 0$. (2) $w_{\mathrm{xyz}}, w_{\mathrm{f}}$ are negatively correlated.

These are very broad conditions, so we have many choices. One of the simplest choices is to use fixed values as a shortcut, such as $w_{xyz} = 1, w_f = 1$. This approach has low computational cost, but it is as inflexible as (1) and (2). On the other hand, we can set w_{xyz} and w_f to be learnable and use a function to normalize the result,

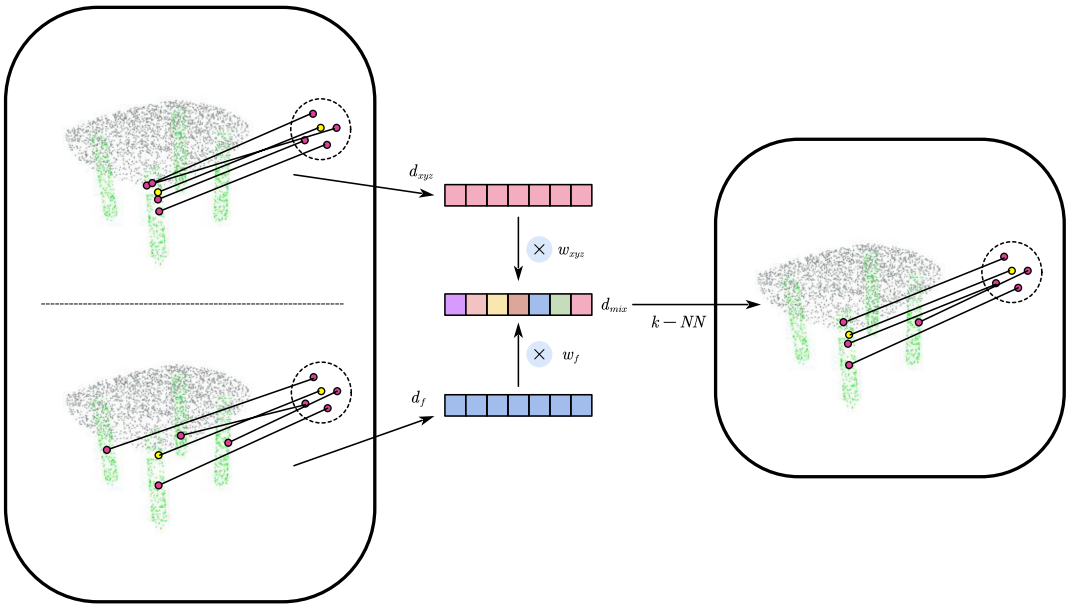

Figure 1. Neighborhood search illustration. Left top: using the distance in 3D space as the search metric. Left bottom: using the distance in feature space as the search metric. Right: The proposed method weights the two metrics by learning from point attention features, then searches the neighborhood using the "mixed" distance. With this method, a point's neighborhood in the point cloud can be flexibly searched.

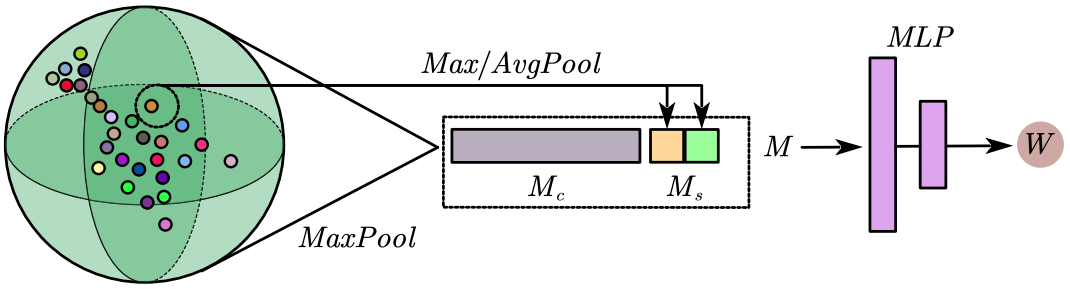

Figure 2. Generation of w_d and w_s from point cloud features. M_c is the global descriptor of the point cloud, and M_s is the point's local feature. The point attention feature is obtained by concatenating M_c and M_s. Then, an MLP layer is used to generate w_d and w_s, and softmax is used to normalize the output.

but this method has difficulty taking advantage of the characteristics of the point cloud itself. Finally, we choose a generation method to obtain w_{xyz}, w_f; specifically, we first obtain the attention feature and then generate the weight from it.

The weight generation step is a two-stage procedure, and is shown in Fig. 2. In the first stage, the attention feature of a point is obtained, and in the second stage, the search weight is obtained from the feature.

$$f_{pa} = \varphi(f) \tag{4}$$

$$w_{xyz}, w_f = \phi(f_{pa}) \tag{5}$$

where $f \in \mathbb{R}^C$ is the point feature and $f_{pa} \in \mathbb{R}^d$ is the attention feature of the point. $w_{xyz}$and$w_f \in \mathbb{R}$ are the search weights.

To obtain the relative features of points, we design a lightweight point attention (PA) operator inspired by Convolutional Block Attention Module(CBAM)[18]. Similar to the CBAM, the PA operator obtains information from channels and space and then generates attention features. The CBAM needs to obtain spatial information from adjacent pixels, but this operation is inefficient in point clouds because point clouds are unstructured. Therefore, the PA operator simply obtains the feature of a single point as the spatial information M_s by $\varphi_1 : \mathbb{R}^{N\times C} \to \mathbb{R}^{N\times d}$, and the channel information M_c is obtained through $\varphi_2 : \mathbb{R}^{N\times C} \to \mathbb{R}^C$. Finally, f_{pa} is generated by concatenating M_c and M_s.

Algorithm. 1 Attention Search Method(ATSearch)

Input: Point Cloud P, Point Features F, Weight Model M
Output: Neighborhood Set Ω

for all p such that $p \in P$ **do**
  $w_{xyz} \leftarrow 0, w_f \leftarrow 0$
  $f_{max} \leftarrow MaxPooling(F)$
  f_p is the corresponding feature of p in F
  $w_{xyz}, w_f \leftarrow M(f_p, f_{max})$
  $D = \emptyset$
  for all $q \in P$ and $q \neq p$ **do**
    f_q is the corresponding feature of q in F
    $d_{mix} = w_{xyz} \times ||q - p|| + w_f \times ||f_q - f_p||$
    add (q, d_{mix}) to D
  end for
  $\Omega_p = mink(D)$
  add Ω_p to Ω
end for
return Ω

Figure 3. Pseudo code for ATSearch.

$$M_s = \varphi_1(f), M_c = \varphi_2(f) \tag{6}$$

$$\mathrm{f_{pa}} = \mathrm{concat}(M_s; M_c) \tag{7}$$

Because φ_2 needs to satisfy the point order invariance, we use *MaxPooling*(*MP*) as the selection. In addition, φ_1 chooses MP with AvgPooling(*AP*).

The pointwise attention feature f_{pa} cannot be used to describe the search weights directly. Therefore, we need to use a function to generate search weights. To define $\phi : \mathbb{R}^d \rightarrow \mathbb{R}, \mathbb{R}$, a simple and logical choice is to generate the initial search weights by learning from f_{pa}; then, we use a function to normalize the weights. In this paper, we use a learnable weight matrix $W \in \mathbb{R}^{2\times d}$ to generate the initial search weights and *softmax* to normalize the search weights:

$$\phi(f_{sub}) = \mathrm{z} \cdot \mathrm{softmax}(Wf_{pa}) \tag{8}$$

And the pseudo-code for ATSearch is shown in Fig. 3.

Applications in point cloud deep learning models. The proposed search method is plug-and-play, so we just need to change the search operation of models to the proposed ATSearch. We integrate the proposed ATSearch into two basic blocks in point cloud deep learning models: SetAbstraction[7] and EdgeConv[11]. the overview of DGCNN after using ATSearch is shown in Fig. 4, because the proposed method is plug-and-play, in order to use ATSearch in point cloud deep learning models, we only need to replace the origin neighborhood search operation with ATSearch in each convolution layer.

Datasets and training details. ModelNet40 dataset[19] and ShapeNetPart dataset[19] are used for training and testing in our study. ModelNet40 dataset includes 12,311 CAD models belonging to 40 categories and ShapeNet part benchmark consists of 16,881 CAD models from 17 categories. Points are uniformly sampled from the CAD model surface. During the training period, parameters were updated by the SGD optimizer, with the learning rate set to 0.1. All experiments are implemented using PyTorch 1.5 and models are trained on one Nvidia RTX Titan.

Results

Tables 1 and 2 show the classification and segmentation performance of different models. PointNet++[7], DGCNN[11], LDGCNN[12], RSCNN[20] are the models in previous works.

In terms of classification performance, instance accuracy is used as the evaluation metric. The performance of each model is improved when the search method is replaced by ATSearch. Among these test models, the best is PointNet++, which has an accuracy increase of 1.1%, and the accuracy improvement of RSCNN is limited because of it has explicitly encoded the points' relations. In summary, the classification results prove that ATSearch can improve the accuracy of existing models without changing their structure.

In terms of segmentation performance, we utilized mIOU as the evalution metric, The result shows that our proposed method is also helpful for segmentation tasks, which shows that our search method can extract enough local details and retain global information compared with the search methods that only depend on spatial distance or feature distance.

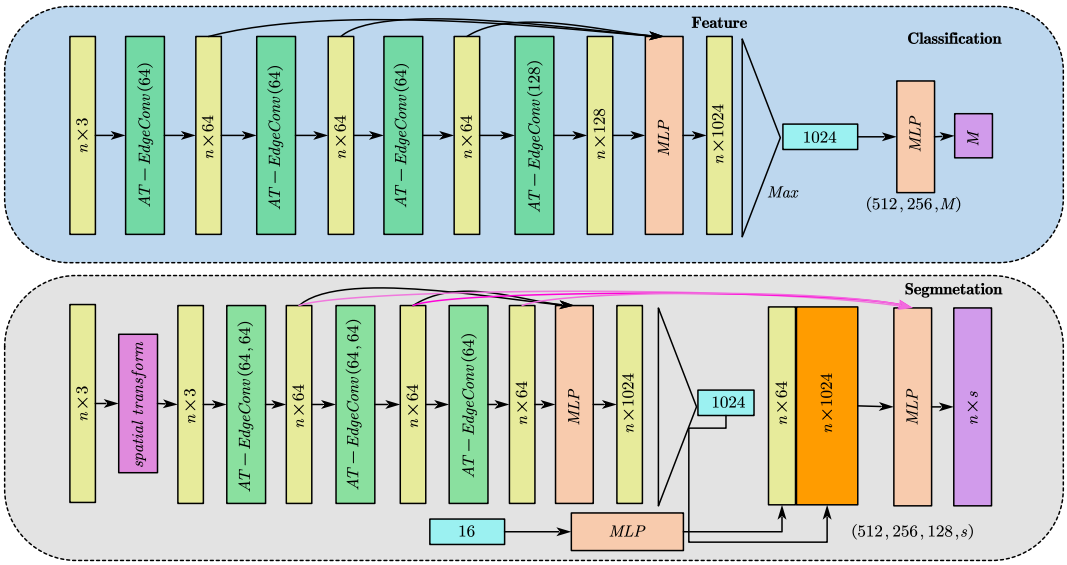

Figure 4. Top: ATDGCNN used for classification task; Bottom: ATDGCNN used for segmentation task. Both of them use the proposed search method in their convolution operator to gather neighbor points.

Model	Accuracy	Accuracy (with ATSeach)
DGCNN	92.4	93.4 (+1.0%)
LDGCNN	92.7	93.0 (+0.3%)
PointNet++	92.1	93.1 (+1.1%)
RSCNN	92.5	92.6 (+0.1%)

Table 1. Shape classification results on Modelbet40 (%).

Model	Instance mIoU	Instance mIoU (with ATSeach)
DGCNN	85.1	85.5 (+0.4%)
LDGCNN	85.1	85.2 (+0.1%)

Table 2. Shape segmentation results on ShapeNet Part (%).

Discussion

To further study the proposed search method, robustness-test, visualization and ablation experiments are performed in this section. The baseline model is chosen to be DGCNN, and the dataset is chosen to be ModelNet40. To show the difference in results when using ATSearch, we name EdgeConv as ATEdgeConv and DGCNN as ATDGCNN.

The choice of the neighborhood size is the key of the search method, it often directly affects the performance and generalization of the deep learning models. Specifically, in the k nearest neighbor algorithm, the selection of k is an important hyperparameter. The compared results are shown in Fig. 5 and ATDGCNN is significantly more robust than DGCNN and LDGCNN.

To explore the influence of the number of our search operations on the accuracy, ablation study is performed and the results are summarized in Table 3 The addition of ATSearch in any layer can improve the performance of the model, but only when the first two layers are equipped with ATSearch together.

There will be a slight performance degradation. This may be caused by the difficulty of matching in the last layer.

T-SNE[21] is utilized to demonstrate the performance of our feature extractor. T-SNE reduces the dimensionality of high- dimensional features to visualize the separability of the features. As shown in Fig. 6, the extracted features are much more discriminative than the original point cloud. Compared with DGCNN, ATDGCNN's output is even more discriminative.

Several point clouds are randomly selected from the ModelNet40 dataset and used to visualize the w_{xyz} and w_f distribution of ATDGCNN's output. It can be seen from Fig. 7 that in different point clouds, the w distribution has its own tendency. For example, the more complex the shape is, the more inclined it is to search the neighborhood

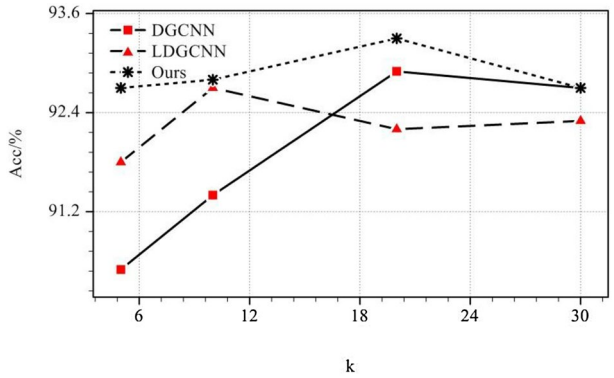

Figure 5. The sensitivity of k used in K-Nearest Neighbor algorithm (k = 5, 10, 20, 30).

Model	AT1	AT2	AT3	Accuracy
Baseline				92.4
A	√			93.2
B		√		93.0
C			√	93.0
D	√	√		92.5
E		√	√	93.1
F	√	√	√	93.4

Table 3. Ablation study of ATSearch on DGCNN(%), AT ***i*** stands for using ATsearch at layer ***i***.

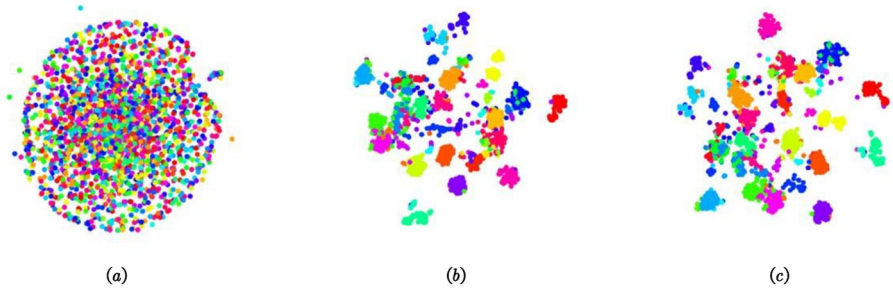

Figure 6. (**a**) Original ModelNet40 dataset. (**b**) DGCNN's output (c) ATDGCNN's output.

in feature space. This shows that the characteristics of the point cloud itself have a unique search preference, so it is proven that our method is feasible.

Finally, to explain how the weights w_{xyz}, w_f affect the model performance, we do experiments with fixed weights: (1) $w_{xyz} = 1, w_f = 0$, the neighborhood is only sampled from 3D space; (2) $w_{xyz} = 0, w_f = 1$. the neighborhood is only sampled in feature space. (3) $w_{xyz} = 0.5, w_f = 0.5$. the neighborhood is sampled from both two spaces with equal probability.(4) w_{xyz} *and* w_f, the neighborhood is adaptively sampled according to the point features from the two spaces. The performances of the DGCNN model with these 4 cases are shown in Table 4 by retraining on ModelNet40 dataset.

As can be seen in the Table 4, compared to the methods of using a neighborhood on a particular space, simply changing w_{xyz}, w_f to 0.5 can improve the performance of the model because the neighborhood can obtain points from both 3D and feature space.

Conclusion

In this study, ATSearch, a plug-and-play search method for point cloud analysis, is proposed. By adaptively combining 3D space searching with feature space searching, ATSearch can flexibly search point cloud neighborhoods, which enhances its ability to obtain information across regions. Moreover, the point attention block has a very low computational cost, so the whole structure is efficient. In the xperiment, our method changes only the model search method to ATSearch and then improves the performance of multiple models. In addition, the proposed

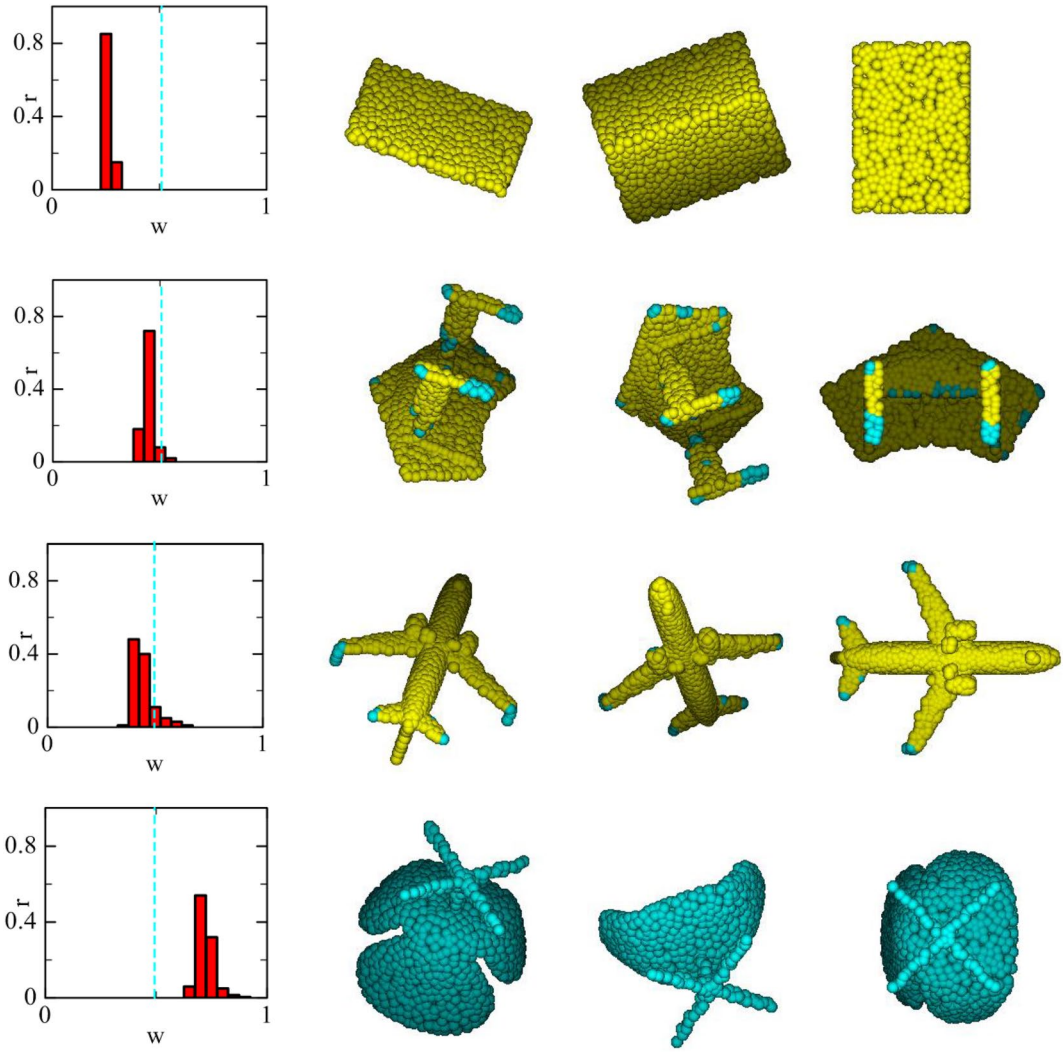

Figure 7. Left: the histogram represents the distribution of search weight in feature space(w_f). Right: In the point cloud, the points in yellow are more likely to search the neighborhood in 3D space ($w_{xyz} > w_f$), while the points in cyan are more likely to search in feature space.

Condition	Accuracy (%)
$w_{xyz} = 1, w_f = 0$	91.7
$w_{xyz} = 0, w_f = 1$	92.4
$w_{xyz} = 0.5, w_f = 0.5$	93.1
ATSearch	93.3

Table 4. Performance comparison under different parameter combinations.

method also shows good robustness to the chosen value of k. Neighborhood search technology is widely used in image processing and point cloud analysis, so ATSearch can be further applied to these fields, such as gesture recognition, emotion classification In the future, it will be worthwhile to consider how to accelerate the inference speed of our proposed network and the form of the weight generation function.

Data availability

All data included in this study are available upon request by contact with the corresponding author.

Received: 6 September 2021; Accepted: 19 January 2022
Published online: 08 February 2022

References

1. Zhou, W., Jiang, X. & Liu, Y. MVPointNet: Multi-view network for 3D object based on point cloud. *IEEE Sens. J.* **19**(24), 12145–12152. https://doi.org/10.1109/JSEN.2019.2937089 (2019).
2. Sun, T., Liu, M., Ye, H. & Yeung, D. Point-cloud-based place recognition using CNN feature extraction. *IEEE Sens. J.* **19**(24), 12175–12186. https://doi.org/10.1109/JSEN.2019.2937740 (2019).
3. Zhou, Y., Dong, H. & Saddik, A. E. Learning to estimate 3D human pose from point cloud. *IEEE Sens. J.* **20**(20), 12334–12342. https://doi.org/10.1109/JSEN.2020.2999849 (2020).
4. Li, M. *et al.* LPCCNet: A lightweight network for point cloud classification. *IEEE Geosci. Remote Sens. Lett.* **16**(6), 962–966 (2019).
5. Zhao, C. *et al.* ALS point cloud classification with small training data set based on transfer learning. *IEEE Geosci. Remote Sens. Lett.* **PP**(99), 1–5 (2019).
6. Zang, Y. *et al.* Density-adaptive and geometry-aware registration of TLS point clouds based on coherent point drift. *IEEE Geosci. Remote Sens. Lett.* **PP**(99), 1–5 (2019).
7. Qi, C. R., Yi, L., Su, H. & Guibas, L. J. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems*, 5099–5108 (2017).
8. Rakotosaona, M.-J., Barbera, La., Guerrero, P., Mitra, N. J. & Ovsjanikov, M. PointCleanNet: Learning to denoise and remove outliers from dense point clouds. *Comput. Graph. Forum* **39**, 185–203 (2020).
9. Bai, X., Luo, Z., Zhou, L., Fu, H., Quan, L. & Tai, C.L. D3Feat: Joint learning of dense detection and description of 3D local features. In *CVPR* (2020).
10. Gao, X., Hu, W. & Qi, G.-J. Graphter: Unsupervised learning of graph transformation equivariant representations via auto-encoding node-wise transformations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 7163–7172 (2020).
11. Wang, Y. *et al.* Dynamic graph cnn for learning on point clouds. *Acm Trans. Graph. (tog)* **38**(5), 1–12 (2019).
12. Zhang, K., Hao, M., Wang, J., de Silva, C. W. & Fu, C. Linked dynamic graph cnn: Learning on point cloud via linking hierarchical features. Preprint https://arxiv.org/abs/1904.10014 (2019).
13. Jiang, M., Wu, Y. & Lu, C. Pointsift: A sift-likenetwork module for 3d point cloud semantic segmentation. Preprint https://arxiv.org/abs/1807.00652 (2018).
14. Li, J., Chen, B. M. & Lee, G. H. So-net: Self organizing network for point cloud analysis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 9397–9406 (2018).
15. Klokov, R. & Lempitsky, V.. Escape from cells: Deep kd-networks for the recognition of 3d point cloud models. In *Proceedings of the IEEE International Conference on Computer Vision*, 863–872 (2017).
16. Shiqi, L., Yu, W., Ke, H. & Shuai, Z. OCNN: Point cloud-based convolutional neural network for object orientation estimation. In *International Conference on Communication and Information Systems (ICCIS)* (2019).
17. Wang, X., Girshick, R., Gupta, A. & He, K. Non-local neural networks. CVPR (2018).
18. Woo, C. S., Park, J., Lee, J.-Y. & Kweon, I. S. Cbam: Convo-lutional block attention module. In *European Conference on Computer Vision (ECCV)* (2018).
19. Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X. & Xiao, J. 3D ShapeNets: A deep representation for volumetric shapes. In *CVPR*, 1912–1920 (2015).
20. Liu, Y., Fan, B., Xiang, S. & Pan, C. Relation-shape convolutional neural network for point cloud analysis. In 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 8887–8896 (2019).
21. Maaten, L. V. D. & Hinton, G. Visualizing data using t-SNE. *J. Mach. Learn. Res.* **9**, 2579–2605 (2008).

Acknowledgements

This work was supported in part by NSFC 11673079, 41801229 and National Key Research and Development Project 2018YFB0505105.

Author contributions

Q.X. and Y.H. conceived the idea, and Q.X. performed simulation. Q.X. analyzed the results with the assistance of Y.H. Q.X. wrote the manuscript and Y.H. supervised the project. D.W. contributes the performance comparison.

Competing interests

The authors declare no competing interests.

Additional information

Correspondence and requests for materials should be addressed to Y.H.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.