Deep Space Probing for Point Cloud Analysis

Yirong Yang, Bin Fan, Yongcheng Liu, Hua Lin, Jiyong Zhang
Xin Liu, Xinyu Cai, Shiming Xiang and Chunhong Pan
yangyirong2018@ia.ac.cn, bin.fan@ieee.org

Problem

3D point cloud is a collection of discrete points which are consist of 3D location information (x, y and z coordinates) and may along with other features. These data are useful in autonomous driving, augmented/virtual reality. However, 3D points distribute in a continuous 3D space irregularly, thus directly adapting 2D image convolution to 3D points is not an easy job. Previous works often artificially divide the space into regular grids, yet it could be suboptimal to learn geometry.

Some Challenges

- GPU memory usage must be affordable.
- The convolution operation should be equipped with multi-weight design.
- All subspaces should not be empty, in other words, making full use of all weights.
- Can be equipped with abundant geometry.
- New convolution operation should be invariant to input permutation.

Memory Analysis

Specifically, let B be the batch size, N_r be the number of local regions of a SPConv layer, N_p be the number of points in each local region. N_k be the number of weights shared by this layer. D_f in and D_{out} be the input and output feature dimension. Suppose $B = 32$, $N_r = 512$, $N_k = 5$, $N_p = 64$, $D_{in} = 64$, $D_{out} = 64$, and the weights and feature maps are stored with single point precision, the memory usage for saving kernels are $B \times N_r \times N_p \times D_{fin} \times D_{fout} \times 4B = 16GB$. To reduce memory usage, many works have to use `sum pooling` function to aggregate proposed features. While we can easily implement our SPConv in a memory efficient way(see the bottom figure of "Pipeline" part), not restricting to the `sum pooling`. We split the feature maps into N_k portions rather than gather kernels for each point. In this way, the memory usage for saving kernels are $N_k \times D_{fin} \times D_{fout} \times 4B = 80KB$. Meanwhile, the memory usage for saving extra feature maps are $B \times Nr \times Nk \times Np \times D_{fin} \times 4B + B \times Nr \times Nk \times Np \times Dfout \times 4B = 2.5GB$. It greatly reduces the memory usage than the naive version.

Network

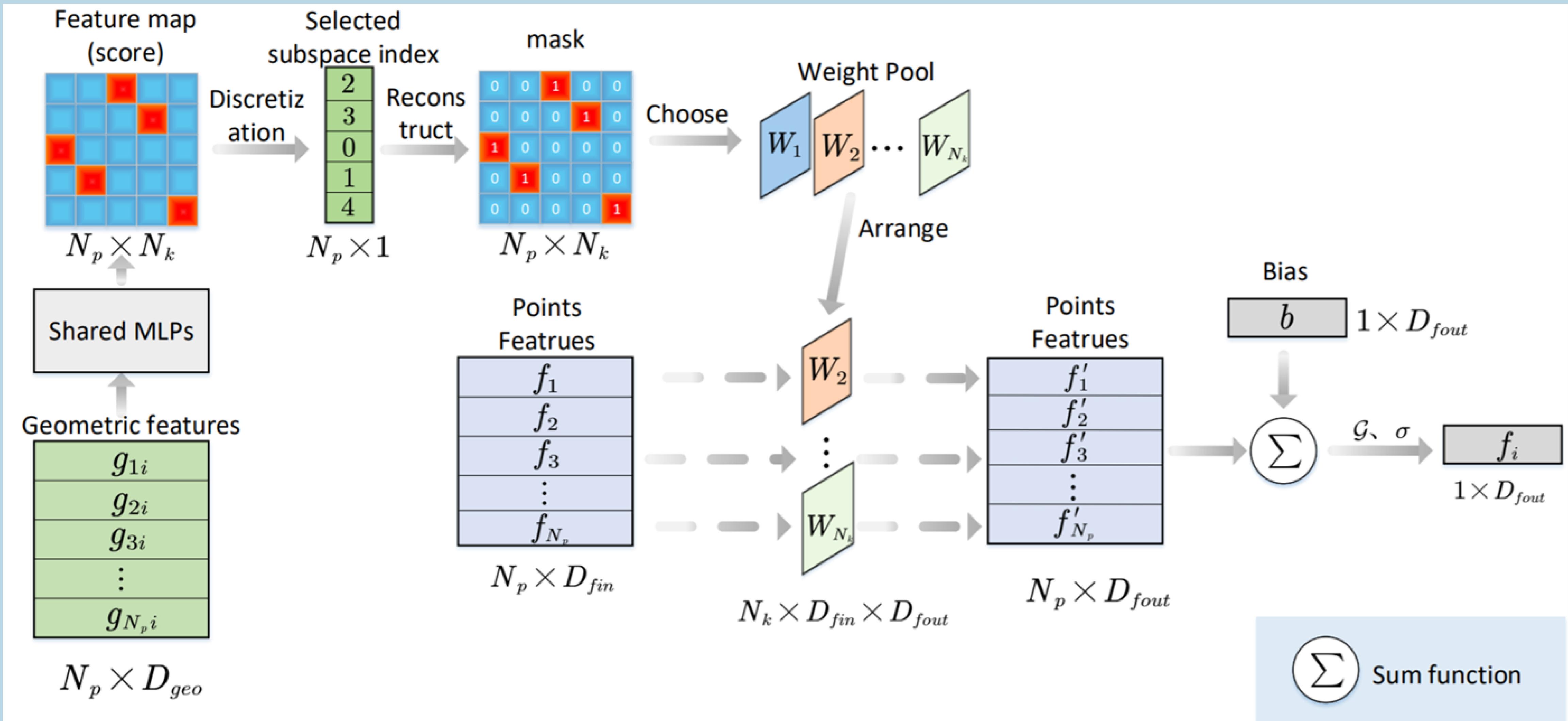
Now that SPConv can sufficiently learn from local features, while point clouds analysis needs global features. We use SPConv as the basis block to build hierarchical neural networks like PointNet++, and we name it Space Probing Convolutional Neural Networks (SPCNN).

Acknowledgements

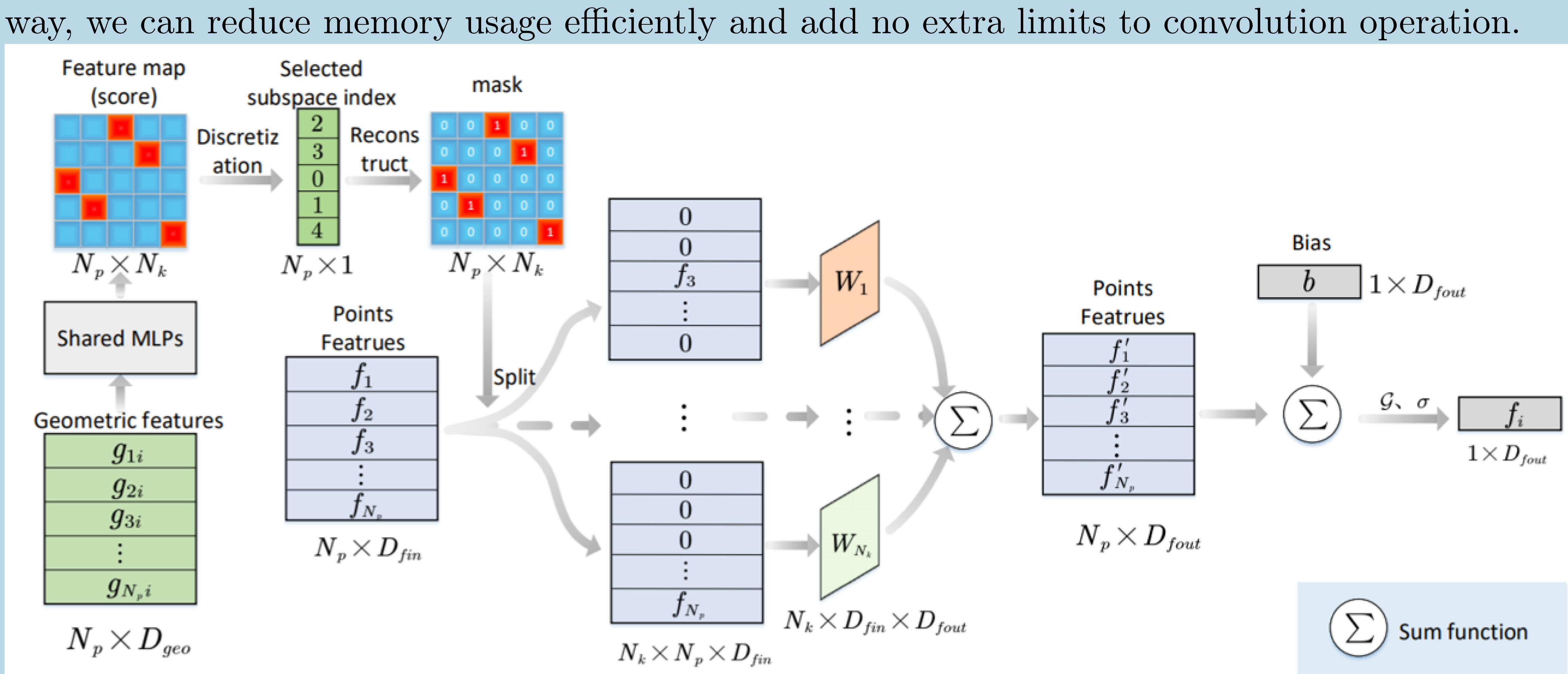
This research was supported by the Major Project for New Generation of AI under Grant No. 2018AAA0100400, the National Key Research and Development Program under Grant No. 2016YFB0501100, the National Natural Science Foundation of China under Grants 61976208, 61773377, and the Young Elite Scientists Sponsorship Program by CAST (2018QNRC001).

The Pipeline of Space Probing Convolution

Our Space Probing Convolution uses points' geometric features as input, and then a following shared MLPs is used to calculate scores, which can be interpreted as the preference between each point and each predefined weight. The *score* matrix can be discrete to a matrix that only has a integer 1 in each line and integer 0 for others. if a integer 1 in the jth column for the ith row, it means the ith point matches the jth weight. A matrix multiplication operation will be implemented between point features and corresponding weight. After that, a aggregate function and a bias will further process the features.

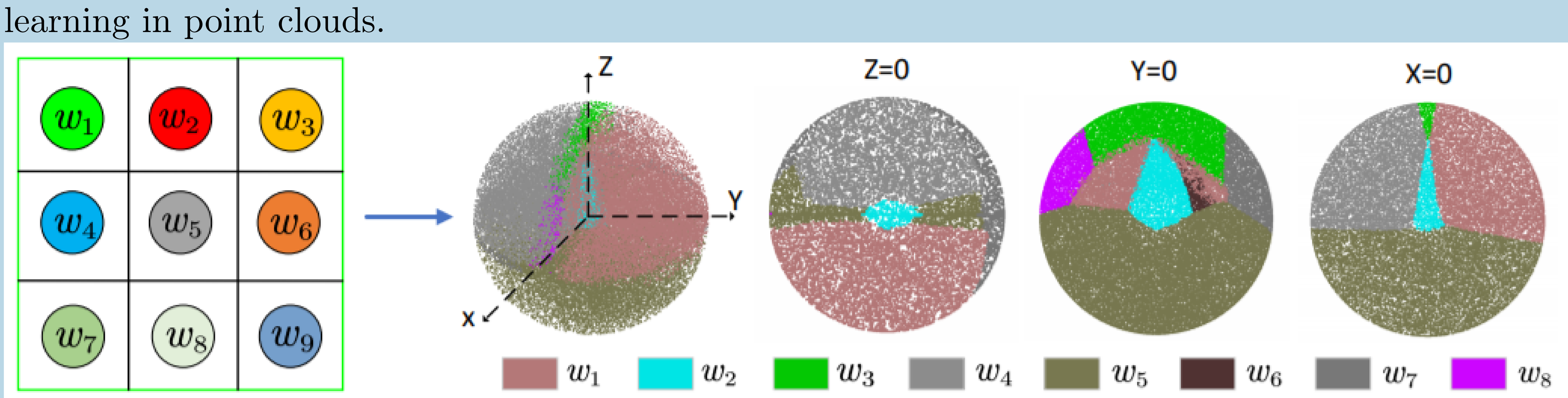


In the naive implementation, each point will get a copy of weight function from the learnable weight function pool, i.e., a group of weight functions, that will cost a large amount of memory. Therefore, we use *index* to reconstruct *mask* matrix, which is the one-hot version of *score* (*index*, *mask* and *score* are variables in the following figure). The points features, denoted as f, will be split into N_k parts, where N_k is the number of weights in the weight pool, by point-wise producting with each row of mask, and each part will be sent to corresponding feature processing network to further process. After that, `sum pooling` function is used to restore the shape and thus obtain new features. In this way, we can reduce memory usage efficiently and add no extra limits to convolution operation.



The Comprehension of Space Probing Convolution

The illustration of image convolution (left part) and our space probing convolution (right part). The image pixels are arranged regularly, since image is a typical grid data structure, thus weight and pixel have well-determined index correspondence. However, these index correspondences are unknown for 3D points, which distribute in a continuous 3D space irregularly. Accordingly, we generalize image convolution by a geometry guided weight selection, which adaptively divides the space for geometric learning in point clouds.



The key to our Space Probing Convolution (aliased as SPConv) is probing the nearby points to adaptively select their corresponding convolution weight from a learnable weight pool. In this way, SPConv adaptively partitions the 3D space into multiple subspaces, and the points located in each subspace share the same weight for convolving features. It convert the grid space to a non-Euclidean space. The right part of this figure is a visualization of space probing results.