PointVector: A Vector Representation In Point Cloud Analysis

Xin Deng* WenYu Zhang* Qing Ding† XinMing Zhang†
University of Science and Technology of China
{xin_deng, wenyuz}@mail.ustc.edu.cn, {dingqing, xinming}@ustc.edu.cn

A. Preliminary

A.1. Problem of WaveMLP.

WaveMLP [11] views the patch of each picture as a wave representation, and considers that the feature of that patch should have two attributes, phase and amplitude, with amplitude representing the actual property of the feature and phase modulating the amplitude that this wave exhibits at a moment. It thus considers that the feature extraction of the patches can be viewed as a superposition of waves. However, there is an important problem, WaveMLP gets an absolute representation of a patch, i.e. the patch is the same when participating in aggregation in any local region. The representation of a patch should be different in different local regions, so we focus on modulating the feature aggregation in local regions. That is, we use a vector representation to better express the relative relationship between neighbor points and centroids in the local region.

In addition, WaveMLP use GroupConv [3] to implement the aggregation and projection process with kernel sizes of 1×7 and 7×1. In this paper we take the form of a combination of the reduction function and GroupConv for aggregation. We give an example of why the original GroupConv is not suitable for this representation of vectors. We take two-dimensional vectors (x_1, y_1) and (x_2, y_2) as an example. The vectors are represented in coordinate form, and then the original vector aggregation method can be formulated as:

$$\begin{aligned} f_{12} &= (w_1(x_1, y_1) + w_2(x_2, y_2)) \cdot (w_3, w_4)^T \\ &= w_3w_1x_1 + w_3w_2x_2 + w_4w_1y_1 + w_4w_2y_2 \\ &= a_1x_1 + a_2x_2 + a_3y_1 + a_4y_2, \end{aligned} \tag{1}$$

where f_{12} denotes the result of aggregating two vectors, w_1 and w_2 are the weights of two vectors in summation, $\{w_3, w_4\}$ is the projection matrix, and a_i is the weight of each component. We can obtain the equation that should be satisfied between the coefficients of each component: $a_1 * a_4 = a_2 * a_3$. That is, the final trained weights need to satisfy this equation for the weighted summation formula of the vectors to hold. However, the network does not impose this restriction on these parameters. So the original groupconv does not preserve the totality of the vector.

A.2. Methodology Review.

The point-based approach was first introduced by PointNet [7]. We denote f_i^{l+1} as the extracted feature of point i after stage *l+1*, N_i as the neighbors of point i and n is the number of incoming points. The simplest point-set operator can be expressed as follows:

$$f_i^{l+1} = R\{H\{[f_j^l, p_j - p_i]\}|j \in N_i\}, \tag{2}$$

where R is the reduction function that aggregates features for point i from its neighbors N_i and H means the shared MLPs.

The subsequent dynamic convolution-based network [12] [17] can be similarly represented as PointNet-like point set operators:

$$f_i^{l+1} = Sum\{\phi\{f_j^l, p_j - p_i\} \cdot f_j^l|j \in N_i\}, \tag{3}$$

where $\phi()$ means the dynamic weight generation function that generates dynamic weights for each point based on the input feature and location information. Eq.3 shows that the reduction function of dynamic convolution chooses sum and uses dynamic weights to generate a new f_j.

Similarly, the attention network [19] can be expressed as a similar point set operator. The core operation can be formulated as follows:

$$f_i^{l+1} = Sum\{att\{f_j^l, f_i^l, pos\} \cdot \sigma\{f_j^l, pos\}|j \in N_i\}, \tag{4}$$

where $att()$ means the attention function that generates attention weights for each point, pos denotes the position information, and $\sigma()$ means the linear transform function without anisotropy. Eq.4 shows that it uses the attention mechanism to update the features of each point j and then uses sum as the reduction function.

Furthermore, template-based methods such as 3D-GCN make use of kernels with relative displacement vectors and

*Co-first authors with equal contribution to refining the theory and experimental design
†Corresponding authors

weights. These weights are influenced by the cosine similarity between the relative displacement vector of the input features and the relative displacement vector of the kernel. The core operation can be formulated as follows:

$$f_i^{l+1} = f_i \cdot kernel_c + \sum_{m=1}^{k} max\{sim\{kernel_m, f_j\} | j \in N_i\},$$
$$sim\{kernerl_m, f_j\} = cos\{dk_m, dp_j\} \cdot kernerl_m \cdot f_j, \tag{5}$$

where k means the kernel size, N_i means the neighbors of point i, $kernel_c$ means the center element of kernel, $cos\{dk_m, dp_j\}$ means Cosine similarity of m-th kernerl element and j-th point feature, $kernel_m$ means m-th kernel element, dk_m, dp_j means displacement vector of m-th kernel element and j-th point feature respectively.

We propose a unique method for generating new features f_j by introducing a vector representation, where the direction of the vector guides the aggregation method.

B. Architecture

B.1. Vector encoder

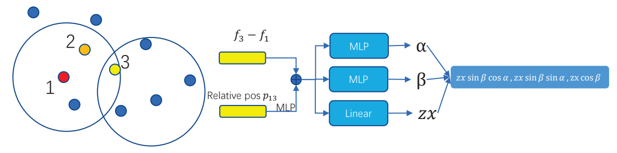
2
3
1
$f_3 - f_1$
Relative pos p_{13}
MLP
MLP
MLP
Linear
α
β
zx
$zx \sin\beta \cos\alpha$, $zx \sin\beta \sin\alpha$, $zx \cos\beta$

Figure 1. **The Vector encoder module.** Two angles are predicted by MLP and zx is transformed by linear.

We provide detailed definitions in the manuscript, and we provide illustrations to illustrate the exact process. As shown in Fig.1, the local information is obtained by a combination of relative features and relative positions. Note that the sum symbol in the figure means sum and ReLU operations. We use the simplest method to predict the angles using MLP, and by default the two angles are independent of each other. For zx, a simple transformation is performed with linear, and then a vector representation is obtained by rotation. The vector representation $v \in R^{B,C\times3,N}$, where B is the batch size, C is the channel of module and N is the spatial size of the input feature of the module.

B.2. Classification architecture.

As shown in Fig.2, we use the LocalVector module to replace the 4 SetAbstract modules and keep the downsampling parameters unchanged. The last SetAbstract was originally used to aggregate all the remaining points, so we leave it as it is. In the classification task, the max reduction fuction has a greater advantage by retaining the most intense part of the variation.

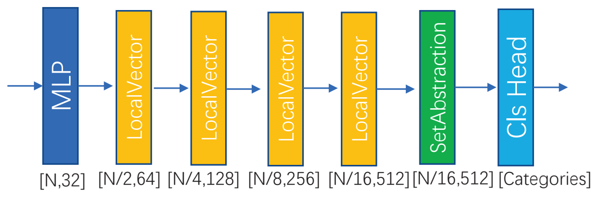

Figure 2. **The Classification architecture PointVector-S.** For comparison with PointNext [9], we replaced the SetAbstract module with the LocalVector module, keeping the other parameters the same.

C. Experiments

C.1. Classification on ModelNet40

Method	mAcc %	OA %
PointNet [7]	86.2	89.2
PointCNN [5]	88.1	92.2
PointConv [14]	–	92.5
KPConv [12]	-	92.9
DGCNN [13]	90.2	92.9
DeepGCN [4]	90.9	93.6
ASSANet-L [8]	-	92.9
Point Cloud Transformer [1]	-	93.2
Point Transformer [19]	90.6	93.7
CurveNet [16]	-	93.8
PointMLP [6]	90.9±0.4	93.7±0.2
PointNet++	-	91.9
PointNet++(PointNext)	89.9 ± 0.8	92.8 ± 0.1
PointNext(C=32)	90.8 ± 0.2	93.2 ± 0.1
PointNext(C=64)	90.9 ± 0.5	93.7 ± 0.3
PointVector-S(C=32)	90.3 ± 0.2	93.2 ± 0.2
PointVector-S(C=64)	91.0 ± 0.5	93.5 ± 0.2

Table 1. Object Classification on ModelNet40.

ModelNet40 [15] is a commonly used dataset for object classification, which is generated by 3D graphic CAD models. It has 40 object categories, each of which contains 100 unique CAD models. Recent works [6] [10] [2] show an increasing interest in the real-world scanned dataset ScanObejectNN [18] than this synthesized 3D dataset ModelNet40. Therefore, we choose to report the results on ScanObjectNN in the manuscript. Furthermore, we report the results of our PointVector-S model on ModelNet40. We use the same parameters as PointNext: CrossEntropy loss with label smoothing, AdamW optimizer, a learning rate of 1e-3, a weight decay of 0.05, cosine learning rate decay, and a batch size of 32 for 600 epochs, while using random scaling and translation as data augmentations. As shown in table 1, the relatively poor performance of our model on the ModelNet40 dataset indicates the limitation of the proposed local vector representation in aggregating global information. We used hyperparameters consistent with PointNext and a training strategy that may not be suitable for our model, which may also account for the relatively poor performance. Note that our network structure on the classification task directly takes vector feature aggregation for downsampling, but max-pooling is probably the simplest and most effective method for downsampling.

C.2. Ablation study

There is a slight problem with the experimental setup in the manuscript, in the 6-fold cross-validation experiment we report the PointVector-L as the standard setup mentioned in the manuscript, but in the S3DIS Area5 and ablation experiments we report the setup of PointVector-L as V=[2, 2, 4, 2]. But, the max+groupconv in the manuscript is reported as V=[2,4,2,2].

Method	size	OA	mAcc	mIOU
		%	%	%
PointVector-L (max+groupconv)	V=[2,4,2,2]	90.6	76.2	70.6
	V=[2,2,4,2]	90.6	77.1	71.1
PointVector-L (sum+groupconv)	V=[2,4,2,2]	90.3	77.21	70.8
	V=[2,2,4,2]	90.8	77.3	71.2
PointVector-XL	V=[3,5,3,3]	90.8	78.3	72.3
	V=[3,3,5,3]	91.0	76.7	71.1

Table 2. Results for models with different number of stagess on S3DIS Area5.

Number of stages. Since the PointVector-L with max+groupconv is reported by another configuration in the manuscript, we compare the two configurations here. As the tab.2 shows, the two reduction functions, max and sum, obtain very similar results, but sum has a higher mAcc and OA. This is consistent with our assumption that better results can be obtained by simply using groupconv to process vectors of each channel independently. Small and large models do not behave consistently in terms of the number of stages. This is an interesting phenomenon, but not the main point of our statement, so it will not be discussed for now.

The following experiments are reported by default as PointVector-XL [3,5,3,3], PointVector-L [2,2,4,2] if no special instructions are given.

Baseline. Our model has some gaps in channel variations and inputs with PointNeXt. To really evaluate whether our model has a greater advantage, we reset a baseline. We take our core operations i.e. Vector encoder and reduction+groupconv+channel mixing Linear was removed and replaced with PointNeXt's MLP+max+MLPs, where the channel of first MLP was transformed from c to 3c. The new model is named PointVector-base. The tab.3 shows that our model has a large improvement in each metric compared to baseline. Also this shows that the other parts of our model are superior compared to the original PointNeXt.

Method	OA	mAcc	mIOU	Params
	%	%	%	M
PointNeXt-XL	90.7	77.5	70.8	41.6
PointVector-base	90.9	77.0	71.4	37.2
PointVector-XL	90.8	78.3	72.3	24.1

Table 3. Baseline. The same experimental configuration was used for all three models.

type	Method	OA	mAcc	mIOU
		%	%	%
feature	$f_j - f_i$+pos	90.8	77.3	71.2
	[$f_j - f_i$,pos]	88.8	70.5	64.9
	f_j+pos	90.9	76.6	70.5
residual	linear	90.8	77.3	71.2
	identity	90.3	75.8	69.3

Table 4. Other Components. $+$ means that the two are added together and then passed through the relu layer. [,] means to directly concatenate two elements.

Other Components. The manuscript mentions that other operations of our model have a larger role, so we conducted ablation experiments on PointVector-L to explore the effect of both input features and residuals on the S3DIS segmentation task. Tab.4 shows that the two parts of the features are added together and then relu can better fuse their information. In addition relative features are more robust than absolute features. The key is that residual uses linear compared to identity, which is a huge improvement.

D. Visualization

As shown in the Fig.3, it can be found that our model performs a little better in complex areas. This shows that we are able to extract more detail in such intensely varied areas than the max-pooling operation of PointNeXt. But we are also prone to miscalculation in flat areas, which is our disadvantage.

E. Code release

Since our model is based on PointNext, we used their code and added a PointVector model. Since our classification and part segmentation and semantic segmentation tasks use different model compositions, the model code is also different, and the corresponding PointVector.py needs to be

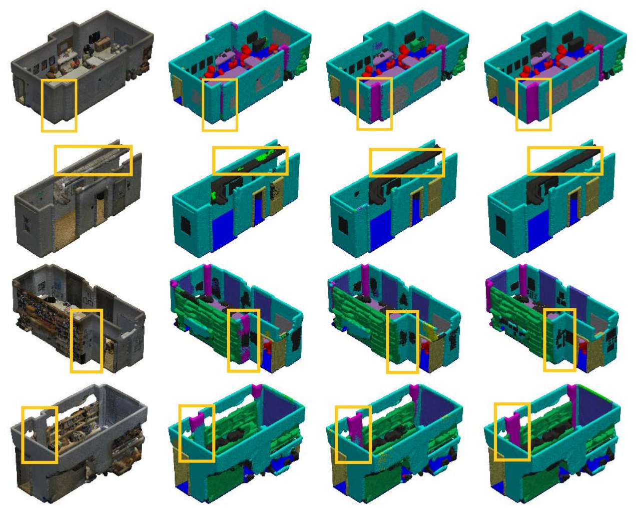

Figure 3. **Qualitative comparisons of PointNext (2^{nd} column), PointVector++ (3^{rd} column), and Ground Truth (4^{th} column) on S3DIS semantic segmentation**. The input point cloud is visualized with original colors in the 1^{st} column. We have circled the different places with a paintbrush.

replaced at runtime. We have not organized the code yet, where PATM represents the core part of our LocalVector module. In the classification and part segmentation tasks, it replaces the convs+max pooling operation in SetAbstraction. See the official instructions for PointNext for related running instructions. And on s3dis our gravity_dim is set to 1. The code of each task is a little different, on ScanObjectNN classification task we insert leakyrelu in the two linear after the reduction function, and the relative features of the input after BN, encoder's activation function all use leakyrelu can reach 88.4% OA, but this is not the main point of our statement, so we do not discuss it for now. The code takes time to organize and we will make it public later.

References

[1] Meng-Hao Guo, Jun-Xiong Cai, Zheng-Ning Liu, Tai-Jiang Mu, Ralph R. Martin, and Shi-Min Hu. Pct: Point cloud transformer. *Computational Visual Media*, 7(2):187 – 199, 2021. 2

[2] Abdullah Hamdi, Silvio Giancola, and Bernard Ghanem. Mvtn: Multi-view transformation network for 3d shape recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1 – 11, Virtual, Online, Canada, 2021. 2

[3] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. 1

[4] Guohao Li, Matthias Mueller, Guocheng Qian, Itzel Carolina Delgadillo Perez, Abdulellah Abualshour, Ali Kassem Thabet, and Bernard Ghanem. Deepgcns: Making gcns go as deep as cnns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021. 2

[5] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. Pointcnn: Convolution on x-transformed points. In *Advances in Neural Information Processing Systems*, volume 2018-December, pages 820 – 830, Montreal, QC, Canada, 2018. 2

[6] Xu Ma, Can Qin, Haoxuan You, Haoxi Ran, and Yun Fu. Rethinking network design and local geometry in point cloud: A simple residual mlp framework, 2022. 2

[7] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, volume 2017-January, pages 77 – 85, Honolulu, HI, United states, 2017. 1, 2

[8] Guocheng Qian, Hasan Hammoud, Guohao Li, Ali Thabet, and Bernard Ghanem. Assanet: An anisotropic separable set abstraction for efficient point cloud representation learning. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 28119–28130. Curran Associates, Inc., 2021. 2

[9] Guocheng Qian, Yuchen Li, Houwen Peng, Jinjie Mai, Hasan Hammoud, Mohamed Elhoseiny, and Bernard Ghanem. Pointnext: Revisiting pointnet++ with improved training and scaling strategies. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. 2

[10] Haoxi Ran, Jun Liu, and Chengjie Wang. Surface representation for point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 18942–18952, June 2022. 2

[11] Yehui Tang, Kai Han, Jianyuan Guo, Chang Xu, Yanxi Li, Chao Xu, and Yunhe Wang. An image patch is a wave: Phase-aware vision mlp. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10935–10944, June 2022. 1

[12] Hugues Thomas, Charles R. Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, Francois Goulette, and Leonidas Guibas. Kpconv: Flexible and deformable convolution for point clouds. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 2019-October, pages 6410 – 6419, Seoul, Korea, Republic of, 2019. 1, 2

[13] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics*, 38(5), 2019. 2

[14] Wenxuan Wu, Zhongang Qi, and Li Fuxin. Pointconv: Deep convolutional networks on 3d point clouds. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2019-June, pages 9613 – 9622, Long Beach, CA, United states, 2019. 2

[15] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE Computer Society Conference on*

Computer Vision and Pattern Recognition, volume 07-12-June-2015, pages 1912 – 1920, Boston, MA, United states, 2015. 2

[16] Tiange Xiang, Chaoyi Zhang, Yang Song, Jianhui Yu, and Weidong Cai. Walk in the cloud: Learning curves for point clouds shape analysis. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 895 – 904, Virtual, Online, Canada, 2021. 2

[17] Mutian Xu, Runyu Ding, Hengshuang Zhao, and Xiaojuan Qi. Paconv: Position adaptive convolution with dynamic kernel assembling on point clouds. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3172 – 3181, Virtual, Online, United states, 2021. 1

[18] Li Yi, Vladimir G. Kim, Duygu Ceylan, I-Chao Shen, Mengyan Yan, Hao Su, Cewu Lu, Qixing Huang, Alla Sheffer, and Leonidas Guibas. A scalable active framework for region annotation in 3d shape collections. *ACM Transactions on Graphics*, 35(6), 2016. 2

[19] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip H.S. Torr, and Vladlen Koltun. Point transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 16259–16268, October 2021. 1, 2