ESANN 2024 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 9-11 October 2024, i6doc.com publ., ISBN 978-2-87587-090-2. Available from http://www.i6doc.com/en/.

Generalizing Convolution to Point Clouds

Davide Bacciu and Francesco Landolfi

Università di Pisa - Department of Computer Science
Largo Bruno Pontecorvo, 3, 56127, Pisa - Italy

Abstract. Convolution, a fundamental operation in deep learning for structured grid data like images, cannot be directly applied to point clouds due to their irregular and unordered nature. Many approaches in literature that perform convolution on point clouds achieve this by designing a convolutional operator from scratch, often with little resemblance to the one used on images. We present two point cloud convolutions that naturally follow from the convolution in its standard definition popular with images. We do so by relaxing the indexing of the kernel weights with a "soft" dictionary that resembles the attention mechanism of the transformers. Finally, experimental results demonstrate the effectiveness of the proposed relaxations on two benchmark point cloud classification tasks.

1 Introduction

Convolutional neural networks (CNNs) [1] have revolutionized numerous computer vision tasks due to their ability to extract hierarchical features from structured grid data like images. However, applying CNNs directly to 3D point clouds, which represent objects as unordered sets of points, proves challenging. Unlike images, point clouds lack inherent structure and ordering, making standard convolution operations inapplicable. This hurdle has motivated researchers to develop specialized convolution approaches for point cloud data. The pioneering work of Qi *et al.* [2] paved the way for deep learning on point clouds. Their approach bypassed the need for traditional convolutions altogether, proposing instead an invariant operator based on set abstraction. Many follow-up works, such as Pointnet++ [3] and DGCNN [4], also proposed different convolution operators on point clouds that share little with the one performed on images or voxel grids. In this paper, we derive instead two relaxations—continuously differentiable approximations—of the convolution operation on grid data that is applicable to unordered sets of points. We achieve this by substituting the indexing of the kernel weights performed by the convolution with a "soft" dictionary that resembles the attention mechanism of the transformers [5]. We test both relaxations and their depthwise variants on two standard point cloud classification datasets, on which they obtain comparable results with respect to common point cloud convolution baselines from literature.

2 Generalizing Convolution to Point Clouds

Given the tensors $\mathbf{X} \in \mathbb{R}^{b_1 \times \cdots \times b_d \times f}$, $\mathbf{Z} \in \mathbb{R}^{(b_1 - c_1 + 1) \times \cdots \times (b_d - c_d + 1) \times g}$, and $\mathbf{K} \in \mathbb{R}^{c_1 \times \cdots \times c_d \times f \times g}$, a d-dimensional multi-channel *convolution* [1] can be defined as

$$\mathbf{Z}_{i_1,\ldots,i_d,k} = \left[\mathrm{conv}(\mathbf{X}, \mathbf{K})\right]_{i_1,\ldots,i_d,k} = \sum_{j_1,\ldots,j_d,h} \mathbf{X}_{i_1+j_1,\ldots,i_d+j_d,h} \cdot \mathbf{K}_{j_1,\ldots,j_d,h,k}, \quad (1)$$

ESANN 2024 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 9-11 October 2024, i6doc.com publ., ISBN 978-2-87587-090-2. Available from http://www.i6doc.com/en/.

where X, Z, and K, represent, respectively, the *input*, the *output*, and the *kernel* tensors.[1] Using a vectorial notation of the indices (all tensor indices start from 0), where $\mathsf{X}_{\mathbf{i}} = \mathsf{X}_{i_1,\ldots,i_d}$ with $\mathbf{i} = [i_1, \ldots, i_d]^\top$, we can rewrite eq. (1) as

$$\mathsf{Z}_{\mathbf{i}} = \big[\mathrm{conv}(\mathsf{X}, \mathsf{K})\big]_{\mathbf{i}} = \sum_{\mathbf{j}\in C} \mathsf{X}_{\mathbf{i}+\mathbf{j}} \cdot \mathsf{K}_{\mathbf{j}}, \tag{2}$$

where $C = [c_1] \times \cdots \times [c_d]$, with $[c] = \{0, 1, \ldots, c-1\}$. Notice that $\mathsf{Z}_{\mathbf{i}} \in \mathbb{R}^g$, $\mathsf{X}_{\mathbf{i}+\mathbf{j}} \in \mathbb{R}^f$, and $\mathsf{K}_{\mathbf{j}} \in \mathbb{R}^{f\times g}$, for all $\mathbf{j} \in C$. Another way to see eq. (2) is by obtaining the output entries by means of the neighbouring ones, that is,

$$\mathsf{Z}_{\mathbf{i}} = \big[\mathrm{conv}(\mathsf{X}, \mathsf{K})\big]_{\mathbf{i}} = \sum_{\mathbf{h}\in N(\mathbf{i})} \mathsf{X}_{\mathbf{h}} \cdot \mathsf{K}_{\mathbf{h}-\mathbf{i}}, \tag{3}$$

where $N(\mathbf{i}) = \{\mathbf{i} + \mathbf{j} \mid \mathbf{j} \in C\}$ denotes the set of neighbouring indices of $\mathbf{i}$.

From eqs. (2) and (3) we can derive two relaxations, that allow us to generalize convolution to (possibly irregular, non-grid-like) point clouds.

First relaxation. Equation (2) can be also represented as a *point cloud convolution*, as follows. Let $V = [n]$ with $n = \prod_{i=1}^{d} b_i$ and assume w.l.o.g. that the convolution is "centered", that is, that the convolution size is odd and uniform across all the dimensions, i.e., $C = \{-c, -c+1, \ldots, 0, \ldots, c-1, c\}^d$ for some constant $c \in \mathbb{N}$, and that the input is properly zero-padded (all these assumption can be replaced by a using a suitable metric and a proper transformation of the indices). Instead of using vector indices, we can vectorize (or "flatten") the tensors up to their d-th dimensions, obtaining $\mathbf{X} \in \mathbb{R}^{n\times f}$, $\mathbf{Z} \in \mathbb{R}^{n\times g}$, and $\mathsf{K} \in \mathbb{R}^{|C|\times f\times g}$, and store instead the d-dimensional vector indices in the auxiliary matrix $\mathbf{P} \in \mathbb{R}^{n\times d}$, such that $\mathsf{X}_{\mathbf{P}_v} = \mathbf{X}_v$ for all $v \in V$. This is a common scenario in point cloud tasks, where $\mathbf{P}$ is used to store the geometric information of the points, also referred as *coordinate matrix*, while $\mathbf{X}$ contains instead other attributes of the points and is generally referred as *feature matrix*. In our setting, similarly to the coordinate matrix, we also define the *offset matrix* $\mathbf{C} \in \mathbb{R}^{|C|\times d}$ containing all the elements of C. We can now reformulate eq. (2) as follows:

$$\mathbf{Z}_v = \sum_{i=0}^{|C|-1} \mathbf{X}_{u^*} \mathsf{K}_i \qquad \text{s.t.} \qquad \mathbf{P}_{u^*} = \mathbf{P}_v + \mathbf{C}_i. \tag{4}$$

We still cannot apply straightforwardly eq. (4) to a generic point cloud, since $\mathbf{P}_u + \mathbf{C}_i$ may not correspond to any point in $\mathbf{P}$. Hence, we can look for the nearest point available, as follows:

$$\mathbf{Z}_v = \sum_{i=0}^{|C|-1} \mathbf{X}_{u^*} \mathsf{K}_i \qquad \text{s.t.} \qquad u^* = \arg\min_{u}\ \delta(\mathbf{P}_v + \mathbf{C}_i, \mathbf{P}_u), \tag{5}$$

[1]Equation (1) should be called "cross-correlation" instead, but in a learning setting it makes no difference whether the kernel is flipped or not.

where $\delta : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is any distance function, such as $\delta(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|$. We can further relax eq. (5) by making it smoothly differentiable by substituting the *argmin* with a *softmin*, as follows:

$$\mathbf{Z}_v = \sum_{i=0}^{|C|-1} \sum_{u \in V} \frac{e^{-\delta(\mathbf{P}_v + \mathbf{C}_i, \mathbf{P}_u)}}{\sum_w e^{-\delta(\mathbf{P}_v + \mathbf{C}_i, \mathbf{P}_w)}} \mathbf{X}_u \mathbf{K}_i. \tag{6}$$

In this way, $\mathbf{C} \in \mathbb{R}^{|C| \times n}$ can also be parametrical, allowing the offsets to be learned adaptively from data. With this last step, we completely replaced the "indexing" performed by convolutions in eq. (2) with an attention mechanism [5], where the *keys* are formed by the point coordinates $\mathbf{P}$, the *values* are the point features $\mathbf{X}$, and the *queries* are instead the neighboring points defined by $\mathbf{P}_v + \mathbf{C}_i$. To avoid computing an attention matrix of size $O(n^2)$, which prevents this formulation to scale to large point clouds, we can "mask" distant points and compute the *softmin* only on the k nearest ones, as follows:

$$\mathbf{Z}_v = \sum_{i=0}^{|C|-1} \sum_{u \in N[v]} \frac{e^{-\delta(\mathbf{P}_v + \mathbf{C}_i, \mathbf{P}_u)}}{\sum_w e^{-\delta(\mathbf{P}_v + \mathbf{C}_i, \mathbf{P}_w)}} \mathbf{X}_u \mathbf{K}_i, \tag{7}$$

where $N[v] = k\text{-NN}(v, \mathbf{P})$ is the k-nearest neighbors of v in $\mathbf{P}$, containing v itself.

Second relaxation. We can derive a similar relaxation by starting instead from eq. (3), as follows. First, we reformulate the definition of neighboring nodes, by adopting instead a closed ball around the "central" node,

$$\mathbf{Z}_v = \sum_{u \in B_c[v]} \mathbf{X}_u \mathbf{K}_{i^*} \qquad \text{s.t.} \qquad \mathbf{C}_{i^*} = \mathbf{P}_u - \mathbf{P}_v, \tag{8}$$

where $B_c[v] = \{u \in \mathbb{N} \mid \|\mathbf{P}_u - \mathbf{P}_v\|_\infty \leq c\}$. Notice that we adopted the infinity norm inside the ball to mimic the "(hyper-)cubic" shape of the convolution's receptive field, but we could also use any other form of neighbourhood such as a 2-norm ball or the k-nearest neighbors. Notice also that, again, we still cannot apply straightforwardly eq. (8) to a generic point cloud, since the offset $\mathbf{P}_u - \mathbf{P}_v$ may not lie in C. We can then apply a first relaxation to eq. (8) by assigning every neighbor displacement to the nearest available offset, as follows:

$$\mathbf{Z}_v = \sum_{u \in B_c[v]} \mathbf{X}_u \mathbf{K}_{i^*} \qquad \text{s.t.} \qquad i^* = \arg\min_i \ \delta(\mathbf{P}_u - \mathbf{P}_v, \mathbf{C}_i). \tag{9}$$

We can further relax eq. (9) by making it smoothly differentiable, as follows:

$$\mathbf{Z}_v = \sum_{u \in B_c[v]} \sum_{i=0}^{|C|-1} \frac{e^{-\delta(\mathbf{P}_u - \mathbf{P}_v, \mathbf{C}_i)}}{\sum_j e^{-\delta(\mathbf{P}_u - \mathbf{P}_v, \mathbf{C}_j)}} \mathbf{X}_u \mathbf{K}_i. \tag{10}$$

In this way, we allow again $\mathbf{C} \in \mathbb{R}^{|C| \times n}$ to be parametrical, so that the offsets are learned adaptively from data. In this case, differently from the previous relaxation, the "indexing" performed by convolutions in eq. (3) is replaced by an attention mechanism, where the *keys* are formed by the offsets in $\mathbf{C}$, the *values* are the filters $\mathbf{K}$, and the *queries* are instead the neighbors' offsets $\mathbf{P}_u - \mathbf{P}_v$.

Depthwise convolution. The two relaxation only affect the way the kernels are averaged together but not do affect instead how they are applied to the feature matrix, i.e., the $\mathbf{X}_u\mathsf{K}_i$ part remains untouched throughout the relaxations. This allows us to generalize the two relaxations also to other kinds of operations, such as the *depthwise convolution* [6]. In the depthwise convolution, defined as

$$\mathsf{Z}_{i_1,\dots,i_d,k} = \left[\text{dw-conv}(\mathsf{X}, \mathsf{K})\right]_{i_1,\dots,i_d,k} = \sum_{j_1,\dots,j_d} \mathsf{X}_{i_1+j_1,\dots,i_d+j_d,k} \cdot \mathsf{K}_{j_1,\dots,j_d,k}, \quad (11)$$

the input and output tensors have the same number of features f, and the kernel tensor has shape $\mathsf{K} \in \mathbb{R}^{c_1\times\cdots\times c_d\times f}$. In the point cloud setting, we have $\mathbf{K} \in \mathbb{R}^{|C|\times f}$ and, if we go over the first relaxation steps again, we obtain

$$\mathbf{Z}_v = \sum_{i=0}^{|C|-1} \sum_{u\in N[v]} \frac{e^{-\,\delta(\mathbf{P}_v+\mathbf{C}_i,\mathbf{P}_u)}}{\sum_w e^{-\,\delta(\mathbf{P}_v+\mathbf{C}_i,\mathbf{P}_w)}} \mathbf{X}_u \circ \mathbf{K}_i, \quad (12)$$

where $\circ$ denotes the element-wise product. A similar formulation can be obtained also for the second relaxation.

3 Related Works

The idea of trainable offsets, even if applied to images, can be dated back to deformable convolution [7]. In their paper, the authors proposed a convolution with variable spacings, similar to our first relaxation, but where the features where aggregated via a bilinear interpolation kernel. In the point cloud literature, instead, we can find several works with a formulation more similar to our second relaxation: in FeaSTNet [8], the authors generalized the convolution by averaging the kernel weights through a (soft-maxed) linear transformation of the local feature vectors, in PointConv [9], the authors averaged the kernel weights via a kernel density estimation, in KPConv [10], via a linear correlation, and in PAConv [11], by using a neural network.

4 Experiments

We tested both relaxations on two standard point clouds classification tasks from literature, ModelNet40 [12] and ScanObjectNN [13]. In the experiments, we used DGCNN [4] as a backbone classifier, where every of its dynamic edge convolution layer was substituted by a relaxed convolution having the same number of channels. The training was performed following the SimpleView [14] protocol: random translation and scaling of the training samples, cross-entropy loss with label smoothing (0.2), and fixed-number of points per sample (1024, with no resampling). As in SimpleView, the final models were obtained after a pre-fixed number of epochs (60), with no feed-back from the test set. To speed up convergence, we used cosine annealing with a warm restart after 20 epochs. In both relaxations we used a k-nearest neighbors with $k = 8$ and we set the number of offsets to 8. These settings were chosen after a preliminary evaluation on a

	ModelNet40		ScanObjectNN	
	Accuracy	Bal. Acc.	Accuracy	Bal. Acc.
SimpleView	93.0 ± 0.4	90.5 ± 0.8	79.5 ± 0.5	— ± —
PointNet	89.2 ± 0.9	85.1 ± 0.6	68.2 ± —	— ± —
PointNet++	92.7 ± 0.1	90.0 ± 0.3	77.9 ± —	— ± —
DGCNN	91.9 ± 0.3	89.1 ± 0.3	78.1 ± —	— ± —
First Relaxation	91.40 ± 0.31	88.15 ± 0.58	72.18 ± 0.81	66.62 ± 1.04
First Rel. (DW)	91.32 ± 0.25	87.57 ± 0.43	71.93 ± 0.75	66.48 ± 1.20
First Rel. (Mobile)	91.77 ± 0.25	88.58 ± 0.48	74.13 ± 0.92	69.28 ± 1.10
Second Relaxation	90.82 ± 0.20	86.53 ± 0.52	67.41 ± 1.11	63.29 ± 1.31
Second Rel. (DW)	90.36 ± 0.35	85.51 ± 0.54	67.92 ± 0.51	64.08 ± 0.91
Second Rel. (Mobile)	90.18 ± 0.28	84.85 ± 0.51	65.16 ± 0.76	57.18 ± 0.98

Table 1: Classification accuracy and balanced accuracy (mean ± std).

validation split (10% of the training dataset). In table 1 we report the average accuracy and balanced accuracy obtained by our relaxations on 10 train/test runs for every model. The relaxations were also tested in their depthwise variant (DW), and in the "inverted residual" variant (Mobile), as proposed in MobileNetV2 [6], were the depthwise convolution is preceded by an expansion layer (6$\times$ channels) and followed by a reduction one. We also restate the accuracies of four common baselines (SimpleView, PointNet/++, DGCNN) as reported in the original paper of the SimpleView training protocol [14]. Missing values were not reported in the paper.

We can see from the results in table 1 that our relaxations produced results comparable to the given baselines. Our relaxation is superior to the lowest performing one (PointNet) almost everywhere, while obtaining a similar result to the backbone model (DGCNN) only when using the first relaxation with the inverted residual setting. Depthwise convolution alone seem to worsen the performance of both relaxations, while it obtained the best results in the mobile setting when combined with the first relaxation. Overall, most baselines obtained better results with respect to our relaxations, which may indicate that their informed inductive bias plays favorably in point clouds tasks compared to one of the general classical convolution.

5 Conclusions

This paper proposes two significant relaxations of the standard convolution operation, specifically tailored for unordered point cloud data. These relaxations address the inherent limitations of classical convolutions, which struggle with the irregular structure of point clouds. The core idea lies in substituting the rigid indexing of kernel weights with a soft attention mechanism. This enables the convolutions to adaptively focus on relevant neighboring points or offsets between points, leading to more robust feature extraction. The experimental evaluation

ESANN 2024 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 9-11 October 2024, i6doc.com publ., ISBN 978-2-87587-090-2. Available from http://www.i6doc.com/en/.

shows that the proposed relaxations achieve comparable performance to existing point cloud convolution methods. This indicates that the relaxations offer a viable alternative framework for applying convolutions to unordered data, while potentially opening doors for further exploration of generalizable convolution operations on such data structures.

References

[1] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning.* MIT Press, 2016.

[2] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, “PointNet: Deep learning on point sets for 3d classification and segmentation,” presented at the Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017.

[3] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, “PointNet++: Deep hierarchical feature learning on point sets in a metric space,” in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., Curran Associates, Inc., 2017.

[4] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, “Dynamic graph CNN for learning on point clouds,” *ACM Transactions on Graphics*, vol. 38, no. 5, 2019.

[5] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is All you Need,” in *Advances in Neural Information Processing Systems*, vol. 30, Curran Associates, Inc., 2017.

[6] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, “MobileNetV2: Inverted residuals and linear bottlenecks,” presented at the Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018.

[7] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei, “Deformable convolutional networks,” presented at the Proceedings of the IEEE International Conference on Computer Vision, 2017.

[8] N. Verma, E. Boyer, and J. Verbeek, “FeaStNet: Feature-steered graph convolutions for 3d shape analysis,” presented at the Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018.

[9] W. Wu, Z. Qi, and L. Fuxin, “PointConv: Deep convolutional networks on 3d point clouds,” presented at the Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019.

[10] H. Thomas, C. R. Qi, J.-E. Deschaud, B. Marcotegui, F. Goulette, and L. J. Guibas, “KPConv: Flexible and deformable convolution for point clouds,” presented at the Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019.

[11] M. Xu, R. Ding, H. Zhao, and X. Qi, “PAConv: Position adaptive convolution with dynamic kernel assembling on point clouds,” presented at the Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021.

[12] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, “3d ShapeNets: A deep representation for volumetric shapes,” in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, ISSN: 1063-6919, 2015.

[13] M. A. Uy, Q.-H. Pham, B.-S. Hua, T. Nguyen, and S.-K. Yeung, “Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data,” presented at the Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019.

[14] A. Goyal, H. Law, B. Liu, A. Newell, and J. Deng, “Revisiting point cloud shape classification with a simple and effective baseline,” in *Proceedings of the 38th International Conference on Machine Learning*, ISSN: 2640-3498, PMLR, 2021.