Supplementary material: Variance-Aware Weight Initialization for Point Convolutional Neural Networks

A Additional experiments

A.1 ModelNet40

We validate our algorithm on the task of shape classification on the ModelNet40 data set [7], composed of only synthetic 3D models.

Data set: This synthetic data set is composed of 12 k shapes from 40 different human-made objects, whereby the shapes are aligned into a canonical orientation. We use the official training/test split which uses 9.8 k shapes for training and 2.4 k shape for testing. We sample points using Poisson disk sampling with a radius of 0.05 from 2 k initial random points on the surface of the objects, which results in roughly 1 k points per model. As features, we use the surface normals, while we use random anisotropic scaling (random scaling in each axis independently between 0.9 and 1.1) for data augmentation. Performance is measured as overall accuracy accumulating the predicted probabilities for each model with different anisotropic scalings.

Network architecture: We used the same architecture as the one used on the ScanObjectNN data set, increasing the number of features by a factor of 2 and decreasing the sampling radius and convolution receptive fields by a factor of 2.

Training: We trained the models using SGD with momentum for 500 epochs, batch size equal to 16, and an initial learning rate of 0.005. We scaled the learning rate by 0.1 after 350 and again by 0.1 after 450, allowing all methods to converge. In order to prevent overfitting, we used a dropout value of 0.2 before each convolution and 0.5 on the final MLP, and we used weight decay loss scaled by 0.0001.

Results: The resulting accuracy is shown in Table 1. Our initialization scheme enables us to eliminate batch normalization while still achieving accuracy values comparable to the standard initialization with batch normalization. In contrast, when using standard initialization without batch normalization, three tested approaches do not learn. This is true for both variants of ours, the "direct" and the "transfer" one, as shown in the last two columns. For the "transfer" one, we computed the initialization on the ScanNet data set [2] for every method on a network with six consecutive layers without any pooling operation. Then, we used this init on the network for the classification task. We see that the performance remains the same as the "direct" setup.

Lastly, we compare our initialization with batch norm when the batch size is reduced. In this experiment, we trained our classification network with and without batch normalization for batch sizes 2, and 4. The results, shown in Table 2, indicate that for a small batch size of 2 our method is able to train the model while methods using batch normalization are not able to converge.

Table 1. Comparison of accuracy obtained with our initialization on different convolution operations, for ModelNet40 Classification.

	BN	NoBN		
			Ours	
	Std	Std	Dir.	Tran.
PCCNN [1]	**91.0**	4.1	90.0	89.7
KPConv [5]	91.1	90.8	91.0	**91.2**
MCConv [3]	**91.0**	4.1	90.4	90.6
PointConv [6]	90.7	**90.8**	90.6	90.1
SPHConv [4]	**90.3**	4.1	90.0	90.0

Table 2. Comparison of accuracy obtained with our initialization against batch normalization for different batch sizes on the ModelNet40 data set.

	2		4	
	Ours	BN	Ours	BN
PCCNN [1]	**90.3**	28.3	90.3	**90.4**
KPConv [5]	**90.3**	14.6	**90.6**	**90.6**
MCConv [3]	**90.0**	14.8	90.0	**90.1**
PointConv [6]	**90.5**	13.2	**90.6**	90.4
SPHConv [4]	**90.0**	19.7	**90.1**	88.6

A.2 Gradient accumulation

We trained the two best performing convolutions on the ScanNet data set, `MCConv` and `KPConv`, with a virtual batch size of 4. We sample each scene as described in paper's Sect. 5.4, i.e., making each scene fill the GPU memory, and we accumulate the gradients over 4 scenes. In this setup, batch normalization still fails, obtaining an IoU of 32.1 for `KPConv` and 54.5 for `MCConv`.

B Implementation details

We implemented the different convolution operations in a unified framework using custom ops to reduce memory consumption. Each method was implemented as a single custom op following the official repositories of the tested operations. However, for some operations, we performed slight modifications. In the following subsections, we will describe the individual changes.

B.1 KPConv

The proposed `KPConv` convolution operation by Thomas et al. [5] use a sum as aggregation method, $\hat{A}_{\text{Sum}}$. However, for the results reported in our paper, we substituted such method by a Monte Carlo estimation, $\hat{A}_{\text{MC}}$. This change was necessary since we experienced instability during training for small batch sizes. We believe that the lack of normalization on the operator produces a high variance in the features between 3D models with dense neighborhoods, such as plants, and 3D models with neighborhoods with low density, such as objects from the class Table. By using Monte Carlo estimation, $\hat{A}_{\text{MC}}$, we were able to obtain a stable training for `KPConv` with low batch sizes. Table 3 presents the accuracy of the original `KPConv` and our modified operation, `KPConv(N)`, for the ModelNet40 data set with varying batch sizes. In these results, we can see that our modification not only makes the training stable for small batches but also achieves higher accuracy than the original operation for large batches. Moreover, we can see that our initialization scheme is also able to train the original `KPConv` operation for large batches.

Table 3. Comparison of accuracy obtained with our initialization against batch normalization for the two variants of `KPConv` on the MODELNET40 data set.

	2		4		8	
	Ours	BN	Ours	BN	Ours	BN
KPConv	4.1	**21.4**	4.1	90.4	**90.9**	91.0
KPConv(N)	**90.3**	14.6	**90.6**	**90.6**	90.6	**91.5**

B.2 PointConv

The `PointConv` convolution operator [6] processes the relative position of the neighboring points with an MLP to produce the weights used to modulate the different features. Unfortunately, when the receptive field of the convolutions is small the relative position is also close to zero, and when the receptive field of the convolution is too large the relative positions are also large. This results in vanishing or exploding gradients for large-scale data sets such as SCANNET. We follow the original implementation and use batch normalization between the layers of such MLP, which normalizes the input based on data statistics and enables training on the aforementioned data sets. However, when batch normalization is not used another approach has to be considered. Other convolutions such as `MCConv` [3] normalize the input by the radius of the receptive field. However, in order to remain as close as possible to the original method, we used instead a custom initialization scheme for the first layer of this MLP:

$$Var(w) = \frac{1}{dr^2}$$

where d is the number of dimensions and r is the radius of the receptive field. Please note that this initialization is only used in our setup when batch normalization is not considered.

C Other architectures

The number of variables on neural network architectures for point clouds is large: convolution operation, pooling method, neighborhood selection, architecture, data augmentation, etc. Each paper uses a different combination of those different variables which not only makes them difficult to compare, but could even mislead the reader on the improvements of novel contributions, such as our weight initialization scheme. Therefore, to strive for reproducibility, we favored a fixed architecture and only modified the convolution operation to see the improvement of our initialization on the different convolutions. However, in this section, we further analyze the effects of our initialization scheme on a different architecture. Moreover, we also compare our architecture with the different architectures proposed for each convolution operator.

C.1 PointConv

In order to measure the improvement of our initialization scheme on other architectures, we used it on the network proposed by Wu et al. [6]. This architecture differs from ours in several parameters. It uses farthest point sampling instead of Poisson disk sampling to reduce the size of the point cloud, it selects the neighboring points using the K-nearest neighbors algorithm, and uses several linear layers between each convolution. We used the code from the official repository and trained a model in the ScanNet data set with batch normalization, a batch size equal to 8, and a point cloud size of $6,000$. Then, we trained the same model without batch normalization and our weight initialization algorithm, but used instead a point cloud size of $16,000$ and a batch size of 1. In order to virtually increase the batch size in our setup, we accumulated the gradients over 8 training steps before modifying the weights of the model. Note that both configurations filled up the memory of our graphics card.

This experiment showed similar results to the ones presented in the main paper. The model with batch normalization obtained a IoU of 57.7, whilst the model without it and our initialization achieved a IoU of 59.3. Moreover, when our weight initialization is not used, the same model is not able to train, achieving an IoU of 1.7. Lastly, the model with batch normalization trained with the larger point clouds and small batch size was able to achieve only an IoU of 30.5.

C.2 Architecture comparison

Lastly, we compare the results obtained on our architecture with batch normalization and without our initialization, to the best results reported on the ScanNet validation set of the different methods. As shown in Tbl. 4 our architecture presents state-of-the-art results, obtaining better or similar results for most of the methods.

Table 4. ScanNet semantic segmentation comparison with our and original architectures, both with standard initialization and batch normalization.

	PCCNN [1]*	KPConv [5]	MCConv [3]	PointConv [6]	SPHConv [4]
Ours	**65.7**	66.0	**66.1**	**64.2**	59.5
Orig.	–	**69.2**	63.3	61.0	**61.0**

$*$ This method was only tested on scenes composed of single models, e.g. ShapeNet.

Bibliography

[1] Atzmon, M., Maron, H., Lipman, Y.: Point convolutional neural networks by extension operators. ACM Trans Graph (Proc. SIGGRAPH) (2018) 2, 4

[2] Dai, A., Chang, A.X., Savva, M., Halber, M., Funkhouser, T., Nießner, M.: Scannet: Richly-annotated 3d reconstructions of indoor scenes. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 5828–5839 (2017) 1

[3] Hermosilla, P., Ritschel, T., Vazquez, P.P., Vinacua, A., Ropinski, T.: Monte carlo convolution for learning on non-uniformly sampled point clouds. ACM Trans. Graph. (Pro. of SIGGRAPH Asia) (2018) 2, 3, 4

[4] Lei, H., Akhtar, N., Mian, A.: Octree guided cnn with spherical kernels for 3d point clouds. CVPR (2019) 2, 4

[5] Thomas, H., Qi, C.R., Deschaud, J.E., Marcotegui, B., Goulette, F., Guibas, L.J.: Kpconv: Flexible and deformable convolution for point clouds. ICCV (2019) 2, 4

[6] Wu, W., Qi, Z., Fuxin, L.: Pointconv: Deep convolutional networks on 3d point clouds. CVPR (2019) 2, 3, 4

[7] Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., Xiao, J.: 3d shapenets: A deep representation for volumetric shape modeling. In: Proceedings of 28th IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2015) 1