

# Efficient Learning on Point Clouds with Basis Point Sets

## Supplementary Material

Sergey Prokudin <sup>\*</sup>  
Max Planck Institute for Intelligent Systems  
Tübingen, Germany  
sergey.prokudin@tuebingen.mpg.de

Christoph Lassner <sup>†</sup>  
Amazon  
Tübingen, Germany  
classner@amazon.com

Javier Romero <sup>†</sup>  
Amazon  
Barcelona, Spain  
javier@amazon.com

### 1. Encoding time

Encoding performance is crucial for real-world applications. Whereas we only state the algorithmic complexity  $O(n \log n + k \log n)$  and note that encoding enables real-time application in the main paper, we provide a detailed analysis of its runtime in this section. For our benchmark, we take a random subset of  $10^3$  ModelNet40 CAD models [10] and sample  $10^1 - 10^5$  points from their surfaces. We also vary the number of the basis points  $k$  used for the encoding. We investigate two implementations of the BPS encoding scheme: the first one uses the  $k$  nearest neighbor implementation available in the sklearn scientific computing package [7]. Here, we use the ball-tree version of kNN search to achieve  $O(n \log n + k \log n)$  complexity of the encoding. This version of encoding can be easily parallelized across multiple CPUs for different point clouds. As an alternative, we also provide a TensorFlow [1] implementation of a direct algorithm for computing pairwise distances. Despite its  $O(kn)$  complexity, it shows remarkable performance even for large values of  $k$  and  $n$  due to the highly efficient execution on a GPU. Various hashing-based algorithms for kNN search designed specifically for 3D data could be used as well [3, 4].

Results are summarized in Fig. 1. Both implementations show super real-time encoding performance for the main application case considered in this paper (*i.e.*, point clouds containing  $n < 10^5$  points encoded with  $k < 10^4$  points). Combined with the proposed deep network, this allows real-time mesh registration from raw scans. Overall, the BPS encoding can be tailored to a specific platform and application via the efficiency-fidelity trade-off in varying  $k$ .

### 2. Space and time complexity

Apart from the FLOPs comparison on the ModelNet40 task, we also provide the inference times for a subset of point cloud processing networks with available implementations and compare them to our models. The results are

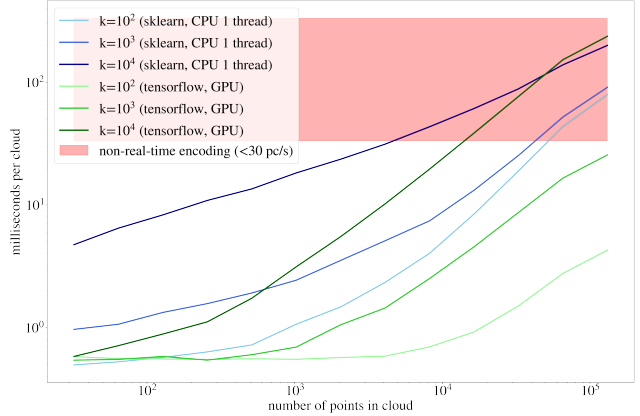


Figure 1. *BPS encoding time.* We measure encoding time per point cloud with respect to the number of input points, basis points  $k$  and the implementation variant. Encoding can be done in real time for the main application cases considered in this paper (point clouds with  $n < 10^5$ , encoded with BPS of  $k < 10^4$ ).

Method	Inference time (CPU)	Inference time (GPU)	Model size
KD-networks [5]	130ms	41ms	150MB
PointNet++ [9]	-	22ms	5.8MB
PointNet [8]	-	8ms	3.2MB
DeepSets [11]	16ms	0.5ms	<b>0.8MB</b>
Ours (BPS-MLP)	<b>0.04ms</b>	<b>0.04ms</b>	1.6MB
Ours (BPS-Conv3D)	56ms	2ms	18MB

Table 1. *Inference time for different models.* Our fully connected model achieve sub-millisecond inference time on both, CPU and GPU.

presented in Tab. 1. Compared to other methods, our fully connected model is able to achieve sub-millisecond inference time even when being executed on a CPU.

### 3. Training details

A complete description of network architectures and training strategy will be made available through publication of our code repository. In this section, we provide a description of the most important elements of the training.

<sup>\*</sup>work was done during internship at Amazon

<sup>†</sup>the last two authors were equally involved

**ModelNet40 models.** For the classification task, our MLP model consists of 2 fully connected hidden layers, each of size 400. Each layer is followed by a ReLU activation and dropout (with probability 0.8 and 0.4 respectively). Class probabilities are given by a final dense layer with a softmax activation. The model is trained with the ADAM optimizer for 2500 epochs with a batch size of 2048 samples, with an initial learning rate of  $1.0e-4$  reduced to  $1.0e-5$  after 2000 epochs. Training data is augmented by few fixed-angle azimuth rotations.

Our Conv3D-model is similar to the original VoxNet [6] architecture, with 4 blocks of  $3 \times 3 \times 3$  convolutions, with each pair of blocks followed by a max-pooling layer of size  $2 \times 2 \times 2$ . This is followed by 2 fully connected layers of size 512 (with separating dropout layers of 0.8 and 0.6). The model is trained with the ADAM optimizer for 505 epochs with a batch size of 512 samples, with an initial learning rate of  $1.0e-4$  reduced to  $1.0e-5$  after 500 epochs.

**FAUST model.** For the FAUST model, the architecture is depicted in Fig. 2 of the main manuscript. There, fully connected layers of size 1024 are used. The model is trained with SGD with momentum of 0.9 for 1000 epochs, with an initial learning rate of 1.0 and followup reductions by a factor of 0.5 every 5 epochs of no improvement on the validation set.

## 4. Results on DYNAMIC FAUST

The efficiency of BPS allows us to process large datasets efficiently. We showcase this in the supplementary video<sup>1</sup>, where thousands of point clouds from the DYNAMIC FAUST dataset [2] are aligned with BPS. We use exactly the same network used in the FAUST experiments without any retraining or fine-tuning; this shows that our network generalizes well to unseen poses and subjects. Each frame is processed independently and no smoothness post-processing was applied to the video. The performance is pretty consistent across the sequences, with one negative factor impacting the accuracy of the alignments: the presence of strong outliers from the floor and the scanner. While the system is robust to the presence of few spurious points around the body, it is sensitive to large amounts of points far from it. Those large chunks drastically change the representation due to the size normalization of the point cloud, which makes the point cloud very different from any cloud in the training set. We believe this could be easily alleviated by introducing similar synthetic noise in our training data or making the normalization more robust to noise.

## References

- [1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pages 265–283, 2016. 1
- [2] Federica Bogo, Javier Romero, Gerard Pons-Moll, and Michael J. Black. Dynamic FAUST: Registering human bodies in motion. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2017*, Piscataway, NJ, USA, July 2017. IEEE. 2
- [3] Bertram Drost and Slobodan Ilic. A hierarchical voxel hash for fast 3d nearest neighbor lookup. In *German Conference on Pattern Recognition*, pages 302–312. Springer, 2013. 1
- [4] Bertram H Drost and Slobodan Ilic. Almost constant-time 3d nearest-neighbor lookup using implicit octrees. *Machine Vision and Applications*, 29(2):299–311, 2018. 1
- [5] Roman Klokov and Victor Lempitsky. Escape from cells: Deep kd-networks for the recognition of 3d point cloud models. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 863–872, 2017. 1
- [6] Daniel Maturana and Sebastian Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 922–928. IEEE, 2015. 2
- [7] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. 1
- [8] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 1(2):4, 2017. 1
- [9] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems*, pages 5099–5108, 2017. 1
- [10] 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 conference on computer vision and pattern recognition*, pages 1912–1920, 2015. 1
- [11] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Ruslan R Salakhutdinov, and Alexander J Smola. Deep sets. In *Advances in neural information processing systems*, pages 3391–3401, 2017. 1

<sup>1</sup><https://youtu.be/kc9wRoI5JbY>